

Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88)

*Sandy Griffin, Editor/Compiler
NASA Lyndon B. Johnson Space Center
Houston, Texas*

Proceedings of a workshop sponsored by the
National Aeronautics and Space Administration,
Washington, D.C., the U.S. Air Force,
Washington, D.C., and cosponsored and hosted
by Wright State University and held at
Wright State University
Dayton, Ohio
July 20-23, 1988



National Aeronautics
and Space Administration

Scientific and Technical
Information Division

1988

PREFACE

This document represents the proceedings of the Second Annual Workshop on Space Operations Automation and Robotics, otherwise known as SOAR '88, which was held at Wright State University on July 20-23, 1988 in Dayton, Ohio.

This workshop was jointly sponsored by the National Aeronautics and Space Administration and the United States Air Force. It was cosponsored by Wright State University. SOAR '88 helped to establish communications between individuals and organizations involved in similar research and technology. It brought together project/program managers in open exchange through presentation of technical papers and panel discussions. The objective of SOAR '88 was to provide a vehicle for engineers, scientists and managers of both the Air Force and NASA to come together in a workshop environment and exchange ideas, problems/problem solutions, and technical information on projects of mutual interest and, perhaps most importantly, to build a solid foundation for future interaction and cooperation. The workshop consisted of technical sessions emphasizing AI/Expert Systems, Human Factors, Environment, Robotics and Application Development and Transition. The workshop will rotate annually between NASA and an Air Force installation.

The papers included in these proceedings were published in general as received from the authors with minimum modification and editing. Information contained in the individual papers is not to be construed as being officially endorsed by NASA.

MESSAGE FROM THE GENERAL CHAIR AND ASSISTANT GENERAL CHAIR

The **SOAR '88** workshop was an outstanding success made possible by the dedication of many Air Force, NASA and contractor personnel.

Our format is continuing to evolve in an effort to best serve the objective of exchanging information among practitioners in common areas of technology application for space operations. We are in the process of quantifying the joint Air Force/NASA savings as a result of exchange of information, software, and initiation of jointly funded projects. I believe a savings of between \$250,000 to \$500,000 (conservative estimate) has been realized as a result of the first annual **SOAR**.

I believe it is essential that we continue to focus **SOAR** on space operations with emphasis on technology that is mature enough to be considered for operational applications. If we do this, we fill a void between the research and development of industry and the Government and the program applications.

Robert H. Brown

SOAR '88 will continue to provide a vital communication link between individuals and organizations involved in space operations and robotics research and technology.

SOAR '88 will present two facets of activities on knowledge-based systems. Sessions will focus on the fundamental issues that must be addressed to continue to advance the state-of-the-art. Additionally, sessions will identify issues addressing transitioning knowledge-based systems to high pay-off applications.

SOAR '88 will also feature sessions in robotics and human factors. These sessions will contribute to the interdisciplinary nature of the workshop. A key event for **SOAR '88** will be the presentations of the National Aerospace Plane and Space Station programs.

We look forward to your attendance and participation. Welcome to Dayton!

Lawrence E. Porter

PRECEDING PAGE BLANK NOT FILMED

ACKNOWLEDGEMENT

Acknowledgements are due to all the personnel who provided the logistic support necessary to the success of this workshop. Thanks are also due to Computer Science Corporation, Lincom, Barrios Technology, McDonnell-Douglas Corporation, and Omniplan Corporation in extending the support of their personnel.

SOAR '88 ORGANIZING COMMITTEE

SECOND ANNUAL WORKSHOP ON SPACE OPERATIONS AUTOMATION AND ROBOTICS

General Chair	Robert H. Brown	NASA/JSC
Assistant General Chair	Lawrence E. Porter	HQ AFSC/USAF
Executive Chair	Sandy Griffin	NASA/JSC
Program Committee	Dr. Thomas Sudkamp	Wright State Univ.
	Maj. Stephen Cross	Wright-Patterson AFB
	Robert Savely	NASA/JSC
	Don Roberts	Griffis AFB
	Dr. Peter Friedland	NASA/Ames Research Cntr
	Lt. Charles Wright	Wright-Patterson AFB
	Dr. Ray Siferd	Wright State Univ.
	Dr. Bob Bachert	Wright-Patterson AFB
	Dr. Anthony Cacioppo	Wright State Univ.
Administrative Committee	Dick Rathbun	Wright State Univ.
	Capt. Brad Van Orden	Wright-Patterson AFB
	Nick Davis	Wright State Univ.

CONTENTS

SESSION 1:	KBS-1 Planning and Scheduling	
SESSION CHAIR:	Capt. James E. Crawford, Wright-Patterson AFB	
	The Scheduling Techniques of ESP2	1
	Space Station Payload Operations Scheduling with ESP2	7
	A Survey of Planning and Scheduling Research at the NASA Ames Research Center ...	15
	An Expert System for Shuttle and Satellite Radar Tracker Scheduling	21
SESSION 2:	KBS-2 Monitoring and Diagnosis I	
SESSION CHAIR:	Chris Culbert, NASA/Johnson Space Center	
	Automation of the Space Station Core Module Power Management and Distribution System	25
	Emma: The Expert System for Munition Maintenance	31
	Diagnostics In the Extendable Integrated Support Environment (EISE)	41
SESSION 3:	KBS-3 Monitoring and Diagnosis II	
SESSION CHAIR:	Chris Culbert, NASA/Johnson Space Center	
	Automatic Detection of Electric Power Troubles (ADEPT)	47
	Real Time Expert System Prototype for Shuttle Mission Control (Paper not provided by publication date.)	51
	Knowledge-Based Operation and Management of Communications Systems	53
SESSION 4:	KBS-4 Engineering Applications	
SESSION CHAIR:	Dr. Les Berke, NASA/Lewis Research Center	
	Nessus/Expert: Bridging the Gap Between Artificial Intelligence and Fortran	59
	Applying AI to Real World Engineering Problems	69
	(Paper not provided by publication date.)	
	A Low Cost Portable Teleautonomous Maintenance Station (Paper not provided by publication date.)	71
	Knowledge-Based Zonal Grid Generation For Computational Fluid Dynamics	73

SESSION 5:	KBS-5 Space Transportation System	
SESSION CHAIR:	Elisha "Rocky" Rachovitsky, Wright-Patterson AFB	
	CIRCA 2000 Operations Criteria	81
	Artificial Intelligent Decision Support for Low-Cost Launch Vehicle Integrated Mission Operations	87
	A Design And Implementation Methodology for Diagnostic Systems	95
	Distributed Artificial Intelligence and Smart Structures in SpaceTransportation (Paper not provided by publication date.)	105
SESSION 6:	KBS-6 Verification and Validation of Knowledge-Based Systems	
SESSION CHAIR:	Eugene L. Duke, NASA/Ames Research Center	
	Application of Flight Systems Methodologies to the Validation of Knowledge Based Systems	107
	Validation Of Highly Reliable, Real-Time Knowledge-Based Systems	123
	Testing Expert Systems	131
SESSION 7:	KBS-7 Advanced Methods in Reasoning with Uncertainty	
SESSION CHAIR:	Thomas Sudkamp, Wright State University	
	Diagnosis by Integrating Model-Based Reasoning with Knowledge-Based Reasoning	137
	Application of Case-Based Reasoning: Design for Manufacturability (Paper not provided by publication date.)	141
	Transformation Based Endorsement Systems	143
	Reducing Uncertainty by Using Explanatory Relationships	149
SESSION 8:	KBS-8 Spaceborne Architectures	
SESSION CHAIR:	Dr. Henry Lum, NASA/Ames Research Center	
	Spaceborne VHSIC Multiprocessor System for AI Applications	153
	Autonomous Satellite Command and Control: A Comparison with Other Military Systems	161
	Garbage Collection Can Be Made Real-time and Verifiable	167
SESSION 9:	KBS-9 Advanced Problem Solving	
SESSION CHAIR:	Don Roberts, Griffis AFB	
	An Architecture for Integrating Distributed and Cooperating Knowledge-Based Air Force Decision Aids	171

Cooperative Problem Solving in Pilots Associate (Paper not provided by publication date.)	177
A Parallel Expert System for the Control of A Robotic Air Vehicle	179
Portable Inference Engine: An Extended CLIPS for Real-Time Production Systems	187
SESSION 10: HF-1 Intelligent Interfaces I	
SESSION CO-CHAIR: Ann Schur, Aerospace and Defense Advanced Technology Laboratory Joseph W. Sullivan, Lockheed Missiles and Space Company	
Browsing Schematics: Query - Filtered Graphs With Context Nodes	193
Presentation Planning Using An Integrated Knowledge Base	205
Adaptation in Human- Computer Interfaces (Paper not provided by publication date.)	219
SESSION 11: HF-2 Metrics of Cognitive Performance	
SESSION CHAIR: Gary B. Reid, Wright-Patterson AFB	
Physiological Assessment of Task Underload	221
A Schema-Based Model of Situation Awareness: Implications for Measuring Situation Awareness	227
Next Generation Keyboards: The Importance of Cognitive Compatability	233
A Method for Modeling Bias in a Person's Estimates of Likelihoods of Events	237
SESSION 12: HF-3 Intelligent Interfaces II	
SESSION CHAIR: Ann Schur, RCA Aerospace and Defense Advanced Technology Laboratory Joseph W. Sullivan, Lockheed Missiles and Space Company	
An Intelligent Multi-Media Human-Computer Dialogue System	245
Knowledge-Based Graphical Interfaces for Presenting Technical Information	253
An Intelligent Interface for Satellite Operations: Your Orbit Determination Assistant (YODA)	259
SESSION 13: HF-4 Intelligent Tutoring Systems	
SESSION CHAIR: Capt. Kevin Kline, Brooks AFB, Tx	
An Intelligent Tutoring System for Space Shuttle Diagnosis	267
Intelligent Tutoring Systems Research in the Training Systems Division: Space Applications	273
Intelligent Tutoring in the Spacecraft Command/Control Environment	279

SESSION 14: HF-5 Knowledge Acquisition for Intelligent Systems
SESSION CHAIR: Joseph W. Sullivan, Lockheed Missiles and Space Company

Conservation of Design Knowledge (Paper not provided by publication date.)	287
Knowledge Acquisition for Case-Based Reasoning Systems	289
Psychological Tools for Knowledge Acquisition	293
Automated Knowledge Acquisition of Design Rationale (Paper not provided by publication date.)	299

SESSION 15: HF-6 Flight Crew Life Support, Protection, and Emergency Escape
SESSION CHAIR: Lanny A. Jines, Wright-Patterson AFB

NASA's Crew Emergency Return Vehicle (CERV) for the Space Station (Paper not provided by publication date.)	301
Light Weight Escape Capsule for Fighter Aircraft	303
Chemical Warfare Protection for the Cockpit of Future Aircraft	309

SESSION 16: HF-7 Intelligence and Telerobotics
SESSION CHAIR: Betty Goldsberry, Lockheed/EMSCO

Optimal Solutions for Complex Design Problems: Using Isoperformance Software for Human Factors Trade-Offs	313
Simulation of the Human Telerobot Interface	321
Human Interactions with Intelligent, Autonomous Systems (Paper not provided by publication date.)	327
Man-Systems Requirements for the Control of Teleoperators in Space	329

SESSION 17: ROB-1 Manipulation and Effectors
SESSION CHAIR: William A. Grissom, Central State University

Integration of a Computerized Two-Finger Gripper for Robot Workstation Safety	335
Local Position Control: A New Concept for Control of Manipulators	339
Dexterity Analysis and Robot Hand Design	343
Concept for A Large Master/Slave Controlled Robotic Hand	353

SESSION 18:	ROB-2 Sensing and Perception I	
SESSION CHAIR:	Timothy Cleghorn, NASA/Johnson Space Center	
	CAD-Model-Based Vision for Space Applications	361
	Decision Tree Analysis of Spacecraft Motion (Paper not provided by publication date.)	369
	Model Based Perspective Projection Expert (Paper not provided by publication date.)	371
SESSION 19:	ROB-3 Task Planning and Reasoning	
SESSION CHAIR:	Jack Aldridge, McDonnell Douglas Astronautics Company	
	Sensor Fusion of Range and Reflectance Data for Outdoor Scene Analysis	373
	Robot Path Planning Using A Genetic Algorithm	383
SESSION 20:	ROB-4 Robotics Teleoperations	
SESSION CHAIR:	Capt. Ron Julian, Wright-Patterson AFB, Ohio	
	Design Concept for the Flight Telerobotic Servicer (FTS)	391
	The WCSAR Telerobotics Test-Bed	397
	The Use of the Articulated Total Body Model as a Robot Dynamics Simulation Tool	403
	Telepresence and Telerobotics	411
SESSION 21:	ROB-5 Supervised Robotics	
SESSION CHAIR:	Brian Wilcox, NASA/Jet Propulsion Laboratory	
	A Novel Manipulator Technology for Space Applications	421
	A Robust Control Scheme for Flexible Arms with Friction in the Joints	429
	Telerobot Operator Control Station Requirements	437
	Time-Delayed Operation of a Telerobot via Geosynchronous Relay	445
	TBA (Paper not provided by publication date.)	449
SESSION 22:	ROB-6 Sensing and Perception II	
SESSION CHAIR:	Timothy Cleghorn, NASA/Johnson Space Center	
	A Representational Framework and User-Interface for an Image Understanding Workstation	451
	Machine Vision for Space Telerobotics and Planetary Rovers	457

Voice Control of Complex Workstations	461
A Multi-Sensor System for Robotics Proximity Operations	467
SESSION 23: ROB-7 Telerobotics	
SESSION CHAIR: Kuldip Rattan, Wright State University	
A Methodology for Automation and Robotics Evaluation Applied to the Space Station Telerobotic Servicer	471
Sensing Human Hand Motions for Controlling Dexterous Robots	481
Application of Model-Based Control to Robotic Manipulators	487
Design Guidelines for Remotely Maintainable Equipment	495
POSTER SESSIONS	
KNOWLEDGE-BASED SYSTEMS	
Expert Systems for Structural Analysis and Design Optimization (Paper not provided by publication date.)	499
Power System Autonomy Demonstration for Space Station (Paper not provided by publication date.)	501
An Expert System for Restructurable Control	503
Adept: An Expert System for Finite Element Modeling (Paper not provided by publication date.)	509
Photonic (Optical) Processors (Paper not provided by publication date.)	511
Neural Networks (Paper not provided by publication date.)	513
Packaging Concepts for Superchip/VHSIC Modules (Paper not provided by publication date.)	515
Expert Systems Applications for Space Shuttle Payload Integration Automation	517
HUMAN FACTORS	
Robotic Telepresence Applications for the Air Force (Paper not provided by publication date.)	525
Ideal: A Methodology for Developing Information Systems	527

ROBOTICS

Model-Based Reasoning for Satellite Autonomy (Paper not provided by publication date.)	533
Development of a Personal-Computer-Based Intelligent Tutoring System	535
SARSCEST (Human Factors)	541
Electronic Data Generation and Display System	553
AUTHOR INDEX	559

THE SCHEDULING TECHNIQUES OF ESP2

John P. Jaap
and
Elizabeth K. Davis

Mission Integration/EL22
Marshall Space Flight Center
Huntsville, Al 35812

ABSTRACT

The Mission Analysis Division of the Systems Analysis and Integration Laboratory at the Marshall Space Flight Center has developed a robust automatic scheduler which can produce detailed schedules for the multi-step activities required for payload operations on the Space Station. This scheduler, a part of the Expert Scheduling Program (ESP2), has five components: the bookkeeper, checker, loader, selector, and explainer. The bookkeeper maintains the usage profiles for nondepletable resources, consumables, equipment, crew, and the times of all the steps for the payload activities for several different schedules simultaneously. The checker searches the data maintained by the bookkeeper and finds times when the constraints of each step of an activity are satisfied. The loader is an expert system which uses the techniques of forward chaining, depth-first searching, and backtracking to manage the workings of the checker so that activities are placed in the schedule without violating constraints (such as crew, resources, and orbit opportunities). The selector has several methods of choosing the next activity for the loader to schedule; new methods using rule-based technology are being studied. The explainer shows the user why an activity was or was not scheduled at a certain time; it offers a unique graphical explanation of how the expert system (the loader) works.

INTRODUCTION

The Experiment Scheduling Program (ESP) has been used to schedule the payload activities of most Spacelab missions and several partial payloads. For Space Station, features are being added to ESP that will automate many of the tasks that now must be done by an expert user. New strategies, rules, and capabilities for scheduling are being developed. A new name, the Expert Scheduling Program, and a new acronym, ESP2, have been adopted to emphasize the improvements being made to the program. These improvements are integrated into this presentation.

Before a description of the scheduling process is given, a few facts about the input data should be stated. The input database to ESP2 contains a mission model, multiple payload activity models each with multiple steps, and the orbit opportunity timelines required by the activity models. The paper entitled "Space Station Payload Operations Scheduling with ESP2" published concurrently with this paper describes the activity modeling capabilities. The program converts the activity requirements to feasibility tests, availability windows, and backtracking rules. The synthesis of the availability windows and backtracking rules into a schedule is the focus of this paper.

THE SCHEDULING PROCESS

When generating a trial schedule, ESP2 first initializes the schedule. Then it begins the "select, check, insert" loop which continues until all requested performances are attempted. The checking process is based on calculating windows which are nested in a hierarchy such that each lower window is totally contained within the window above it. At each level of the hierarchy, a window may contain many windows at the next lower level, and each of these may contain many windows. Checking of constraints begins at the topmost window and proceeds downward and across. When an acceptable window is found, checking proceeds immediately to the next lower level. If, at any level, an acceptable window cannot be found, the window above it is failed, and the next window at the level of the failed window is defined. This technique is a micro-example of depth-first searching. Checking is successfully completed whenever an acceptable window is found at the bottom level. Checking fails whenever another window at the top level cannot be defined. Models with two-way concurrency (mandatory concurrency) are processed simultaneously. There are eleven different types of windows within the hierarchy. Five of these apply to the entire performance and six apply to each step of the activity model. The windows for each step are independently nested; i.e., the crew subwindow for step 5 of a model is

nested within the nondepletable/equipment subwindow for step 5, but is not nested within any of the windows for the other steps on that model. If a model or step does not have a particular requirement, the associated window is set equal to the next higher level window.

The scheduler is described by presenting on a step-by-step basis how an expert mission planner would do the task manually and then by showing how this approach is implemented in ESP2. Since the scheduler emulates how an expert would do the task manually it meets the primal definition of an "expert system." When explaining the scheduling process, it is convenient to start at the bottom (the bookkeeper) and work upwards to the selector.

Figure 1 shows the major components of the program (only ESP) and their interfaces. While the primary purpose of the bookkeeper and the checker is to support the loader, they also support other features of ESP2 such as retrieving a previously generated schedule from a file, schedule editing, schedule validating, and automatically resolving conflicts.

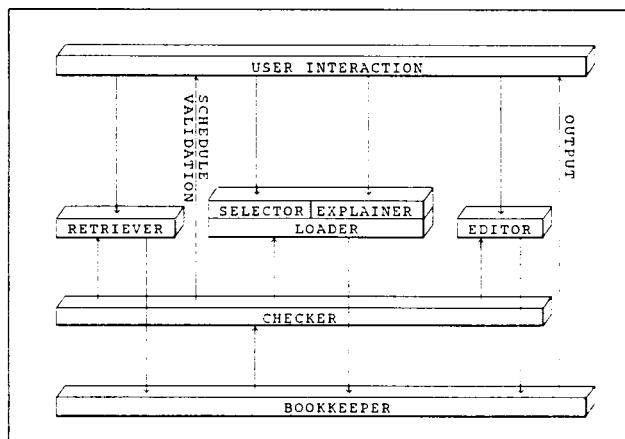


Figure 1. Major Components & Interfaces

THE BOOKKEEPER

Assume that a mission planner wants to schedule some activities which require power. Also assume the planning period is one week and the available power is 10.0 kilowatts. The planner would initialize a table like the one shown in Figure 2 with the initial time and value and the final time and a value of zero. The table is read: beginning at time 0/00:00:00 (zero days, zero hours, zero minutes, zero seconds), the power available is 10.0 kilowatts; at 7/00:00:00, the power available changes to 0.0 kilowatts.

Suppose that an activity is scheduled from 1/13:45:00 to 3/16:00:00 using 2.5 kilowatts of power. Also suppose that another activity is scheduled from 4/03:28:45 to 4/09:00:15 using 3.75 kilowatts of power.

Four new times are inserted into the table showing the reduction and restoration of power by each activity.

The process becomes more intricate when activities overlap each other. Assume that another activity is added from 3/16:00:00 to 5/18:35:30 using 1.25 kilowatts of power. Since the start time already exists in the table, it is only necessary to insert one new time. However, the power availability for all the time points between the start and stop times must be updated.

Initial Time Power	With 2 inserts Time Power	With 3 rd insert Time Power
0/00:00:00 10.00	0/00:00:00 10.00	0/00:00:00 10.00
7/00:00:00 0.00	1/13:45:00 7.50	1/13:45:00 7.50
	3/16:00:00 10.00	3/16:00:00 8.75
	4/03:28:45 6.25	4/03:28:45 5.00
	4/09:00:15 10.00	4/09:00:15 8.75
	7/00:00:00 0.00	5/18:35:30 10.00
		7/00:00:00 0.00

Figure 2. Sample Bookkeeper Table

This example is lacking only in scale! Doing the task manually would require keeping up with dozens of resources and thousands of events. In addition to keeping up with resource and crew usage, it is also necessary to keep up with the activity schedule so that sequencing, concurrency, and performance delays can be checked, and to keep up with crew location so that crew translation time can be allotted. Notice that the length of the table is a function of the number of resource level changes, not the duration of the flight increment being scheduled.

The bookkeeper in ESP2 emulates the manual process described above by using specially designed file formats and processing techniques to rapidly access and update the data. Updating all the intervening time points between the start and stop times of an activity consumes time but permits the checker to operate much more efficiently. Since ESP2 is implemented on a 32-bit machine and time is maintained in integral seconds, the flight increment length is limited to the largest integer that can be stored in a 32-bit word; i.e., 2,147,483,654 seconds or 68+ years.

Space Station payload activity scheduling philosophy calls for scheduling models or groups of models within resource allocation envelopes. In this case, the bookkeeper is pre-loaded from a database containing the envelopes.

THE CHECKER

The data from the bookkeeper and the model requirements can be used to derive availability windows for most model requirements. A mission planner performing the checking function might want to know, "where does the available power first exceed 8.00 kilowatts?" A quick scan of the table in Figure 2 would determine that the window opens at 0/00:00:00 and closes at 1/13:45:00.

ORIGINAL PAGE IS OF POOR QUALITY

If the next window were needed, the mission planner would remember how far down the table checking had proceeded, resume there and find that the next window runs from 3/16:00:00 to 4/03:28:45, followed by the window from 4/09:00:15 to 7/00:00:00.

In ESP2, the request presented to the checker might be, "Give me the first window where all the resources required for my activity are available." The request would also contain earliest and latest times of interest. This limiting of the search range allows the checker to respond much faster. The rules for converting the model requirements to windows reside within the checker. It can respond to questions about availabilities of orbit opportunities, equipment, nondepletable resources, or crew. It can also respond to queries about the windows where performance delay, sequencing, and concurrency requirements are met. Allowing time for crew movement from one location to another is accomplished by subtracting the translation time from each end of the crew availability windows. A matrix of translation times is provided in the database.

THE LOADER

The first step in describing how the loader works is to describe how a mission planner would load a single step into the schedule. After checking to see that the consumables were available, the mission planner would determine the window in which the step might be scheduled. The planner would then choose one of the requirements and find the first window satisfying that requirement. If this window were long enough, the mission planner would check for the window of another requirement. Checking would continue in this manner until all requirements were met. If the intersection of the windows were as long as the step, the step could be scheduled. In ESP2 the loading, or scheduling, of steps is accomplished in a manner similar to the human process described above. The requirements are checked in a particular order; i.e., the windows have a defined hierarchy. Each window is the search range for determining the window below it in the hierarchy. Whenever a window is unacceptable (shorter than the minimum step duration), the next window in the same search range (next higher window) is requested. Whenever no usable window is found at a particular level, the window above it becomes unacceptable and the next window at that level is checked. Checking continues in this manner until a set of nested windows for all requirements has been generated or there are no more windows at the highest level and the step cannot be loaded. The reader has probably noticed that the bulk of the summary given in a preceding section was merely a synopsis of the loader.

Once a lowest-level acceptable window is found, the step is loaded within this window according to several rules. The highest priority rule is to start the step as early as possible (front load). Another rule is to maximize the step duration within the

window up to the limit specified on the model. If after assigning the start and stop times of the step there remains a choice of crew members to assign, then they are assigned so as to balance the crew usage.

But the task is not just to schedule steps, but to schedule performances of multi-step models! This task requires finding a place in the schedule which meets not only the requirements of all steps individually, but also the delay constraints between steps and performances, the resource carry-through requirements, the sequencing and concurrency requirements, and the performance windows specified with the model.

Confronted with this larger task, the mission planner would begin by defining a window in which the performance must be scheduled. This window is determined by considering the window from the selector, the performance window from the model, performance delays, and sequencing. The mission planner would load the steps in this window on a trial basis, moving them around until all steps were validly loaded. Only then would the bookkeeping function be conducted and the performance actually scheduled. A detailed example of front loading a performance is given in Figure 3.

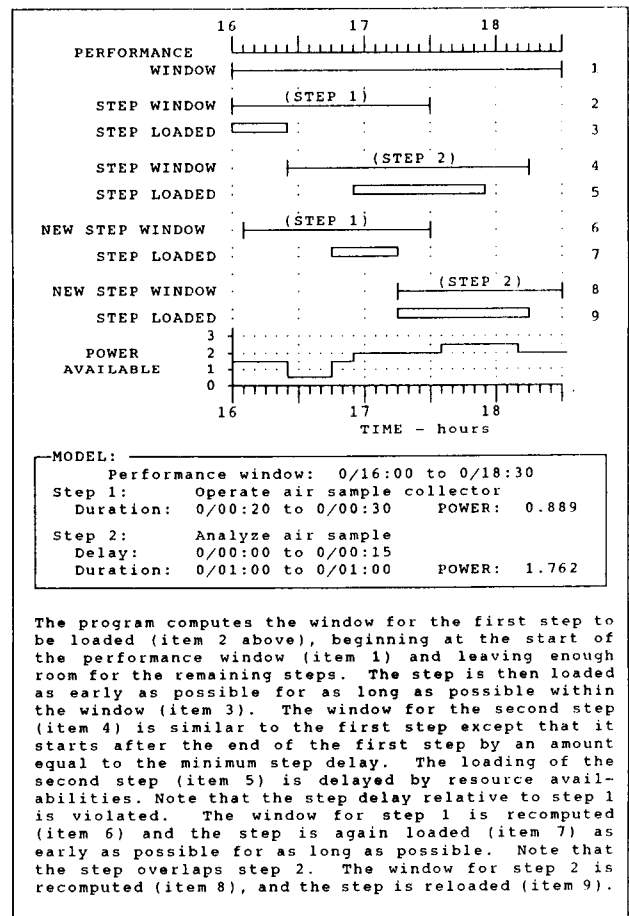


Figure 3. Performance Loading with Backtracking

Model requirements such as the delay between steps, resource carry-through, crew lockin, maximum performance duration, and others are implemented as backtracking rules in the loader. Firing one of these rules results in already-loaded steps being reloaded. When reloading steps, the program adjusts the step window so that the same backtracking rule is not violated again.

The above discussion does not take into consideration all of the requirements that exist in the domain specified for ESP2. Other rules are included to check for consumables. If consumable usage is a function of time, the step durations may have to be chosen at less than the otherwise available maximum. The scheduling and descheduling of startup and shutdown steps must be performed and crew monitoring must be scheduled.

After all steps of a performance are loaded, the performance is scheduled and all bookkeeping functions are conducted. In addition to asking the bookkeeper to update its data, the loader also updates the crew balancing parameters, the grading equation parameters and other data.

ESP2 does one more thing that a good mission planner would do. After scheduling a performance of a model, the program saves a snapshot of its computed data. When asked to schedule another performance of that model, the program resumes scheduling based on the snapshot. This heuristic feature significantly enhances the response time. However, it is possible for a snapshot to be invalidated by the scheduling of another model. Therefore after scheduling each performance, ESP2 checks all snapshots and clears those that are invalid.

This is only half of the story! When scheduling models with two-way concurrency (neither model can be scheduled without the other), ESP2 processes them simultaneously. The complexity of the loading task is at least doubled. Since both models are active at the same time and may require the same resources, the checker must account for the already loaded steps of the other model when computing availability windows. Determining when and how to backtrack is more complex and the computation of step windows is more difficult.

The loader has a last-chance ploy that is invoked after all other attempts to schedule a performance have failed. If any steps have variable durations, these durations are forced to the minimum value and the loading process is repeated. Remember, the original desire was to maximize step durations.

The preceding description applies to front loading; i.e., loading the performance as early as possible. Back loading (loading the performance as late as possible) is the mirror image of front loading.

THE SELECTOR

The loader places the models in the schedule based on model requirements and current availabilities. Since the program cannot deschedule or reschedule a performance (global backtracking) while generating a schedule, the order of attempting to schedule the models, referred to as the selection order, has a significant effect on the schedule. Selecting the next model/performance to schedule is the function of the selector. The selector also determines the topmost window in the performance-level hierarchy, the steps to be scheduled (the model scenario), and the loading algorithm.

The selector may specify the topmost window so as to force the loader to place the performance as required by a particular scheduling strategy. Possible scheduling strategies are resource usage leveling, reserving certain times for other yet to be scheduled activities, etc.

As part of the input database, the user specifies multiple scenarios for executing a performance of a model and the value, or weight, of each. Normally the selector requests that the loader attempt to schedule the highest-valued scenario; and, if it fails, the selector requests the next lower scenario; and so forth. Alternately, the user can specify a selection method that selects an equal number of each scenario or selects them proportional to the values.

The user also specifies which loading algorithm to use or specifies that the selector itself is to choose the algorithm. But most importantly, the selector chooses which model to schedule next. The user divides the models to be scheduled into groups, assigns a selection method to each group, and then specifies the order in which to process the groups. The selector processes the groups and passes the models to the loader one performance at a time. The four most commonly used selection methods are described below.

The fixed-order method allows the user to specify the complete order of selecting the models, possibly on a performance-by-performance basis. The user also specifies the rules for selecting the model scenario. When processing a fixed-order group, ESP2 makes automatic adjustments to account for sequencing and concurrency. ESP2 supplies a command which can reorder the members of a fixed-order group so that the most difficult to schedule is selected first. This command considers the time windows, the orbit opportunity requirements, and the number and duration of performances requested by the model.

The random-order method requires the user to specify a seed for a pseudo-random number generator, the group members, relative weighting factors for the members, and the scenario selection rules. As it processes a random-order group, ESP2 takes into account sequencing and concurrency. At the user's request, ESP2 automatically generates

ORIGINAL PAGE IS OF POOR QUALITY

multiple schedules and saves the "best" one. This Monte-Carlo technique allows the program to perform exhaustive searches without further user intervention.

The maximize-grade method allows the user to specify the members of the groups, the weighting factors for each of the schedule grading parameters, and special scenario selection rules. The scheduling order is determined by the selector based on which model (if scheduled) would yield the maximum increase in the grade.

The program also provides a pseudo or manual selection method. The scheduler editor provides a gateway to the automatic scheduler which bypasses the selector. In this mode the user performs the task of the selector — selecting the model and specifying the scenario and the topmost window.

THE EXPLAINER

As a companion to the loader, ESP2 provides a package which traces the step-by-step activities of the loader. This package can help the user understand why the program could not schedule another performance of a model or why it scheduled the performance where it did. The program usually cannot tell why a model cannot be scheduled. Indeed, there is often not a single reason. The crew may be available when the orbit opportunity is not, or the orbit opportunity may be available when the power is not, etc. No one reason can be stated for the failure. The trace package can show (literally) why the performance could not be scheduled. In order to interpret the trace, the user only needs a basic understanding of the scheduling process.

Primarily, the trace shows the windows returned by the checker and the firing of backtracking rules. The data delivered by the selector and feasibility test failures are also shown. The text presentation contains messages like the following:

- "resource window from 17/23:45 to 18/3:30", — window returned from the checker,
- "end of window checking crew", — the checker could not find a window within the specified search range (next-higher window),
- "window unacceptable", — window is shorter than the minimum step duration,
- "step 4 scheduled from 18/1:25 to 18/2:25", — step is tentatively loaded,
- "delay from step 3 to step 4 violated", — the loader must backtrack and reschedule step 3
- "chosen crew: 1,3", — crew members 1 and 3 are assigned to the step, and
- "minimizing" — the loader has invoked the last-chance rules.

These are only a sampling of the 74 possible messages.

The trace display is divided into two windows; one window containing the text and the other the graphics. Only the messages

containing times (window and scheduled messages) are plotted. Figure 4 shows a typical trace display for a model without two-way concurrence. When models with two-way concurrence are scheduled, both are loaded and traced at the same time. Then both sets of performance level windows are shown above the dotted line. Since the loader only processes one step at a time, only one is shown below the dotted line.

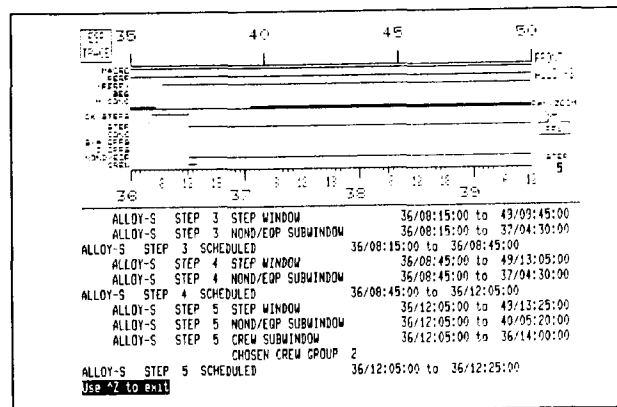


Figure 4. Typical Trace Display

The scale of the performance level portion of the plot is determined by the topmost window. The user may pan and zoom in the bottom (the step level) portion of the plot. The user may also scroll forward and backward through the messages. As each window or scheduled message is presented in the text window, it is plotted in the graphics window.

SUMMARY

In the Expert Scheduling Program all model requirements are reduced to feasibility tests, windows, or backtracking rules. Which model/performance is scheduled next is determined by the selector. The checker calculates the windows, and the loader combines the windows and executes the backtracking rules. The explainer presents the user with a trace of the loading process.

This technique has been proven over the last ten years by scheduling Spacelab payload activities. Only two conditions are necessary for it to support Space Station payload activity planning: the activity requirements must be reducible to a combination of feasibility tests, windows, or backtracking rules; and robust selection methods must be found to eliminate the now-required assignment of selection methods by an expert user.

Surveys of future payload descriptions have uncovered only a few requirements which are difficult to express in the required formats; for example: after-effects of an activity require including a step in the model which uses the affected resource.

The portfolio of selection methods in ESP2 is easy to expand and many additional scheduling strategies can be implemented as selection methods. Several additional selection methods have been designed and are being implemented. An expert system for dividing the payload activities into groups and choosing the best selection method (scheduling strategy) for each is being considered.

There is a parallel effort within our group and the community at-large to formulate a different (and better) method to schedule Space Station payload activities. As with any research effort, there is no guarantee that the research will be successful, especially within the limited time span available before Space Station will become a reality.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Mr. Jerry Weiler, the scheduling expert whose guidance was invaluable.

BIBLIOGRAPHY

1. Britt, D.L.; Geoffroy, A.L.; Schaefer, P.R.; Gohring, J.R.: "Scheduling Spacecraft Operations", Conference on Artificial Intelligence for Space Applications, Huntsville, Al., Nov. 13-14, 1986.
2. Deuermeyer, B.L.; Shannon, R.E.; Underbrink, A.J., Jr.: "Creation of the Selection List for the Experiment Scheduling Program (ESP)", Industrial Engineering Division, Texas Engineering Experiment Station, Texas A & M University, College Station, Texas. Final Report of NASA Contract No. NAS8-35972, NAS8-1778861, May, 1986.
3. Kurtzman, C.R.: "Time and Resource constrained Scheduling, with Application to Space Station Planning", Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Ma., 1988
4. Maxwell, T.G.: "Investigation of Experiment Selection Order Determination for Space Mission Scheduling", M.S. Thesis, University of Alabama in Huntsville, 1987.
5. Stacy, K.L.; Jaap, J.P.: "Space Station Payload Operations Scheduling with ESP2", Second Annual Workshop on Space Automation and Robotics (SOAR 88), Dayton, Ohio, July 20-23, 1988.

SPACE STATION PAYLOAD OPERATIONS SCHEDULING WITH ESP2

Kenneth L. Stacy
and
John P. Jaap
Mission Integration/EL22
Marshall Space Flight Center
Huntsville, Al 35812

ABSTRACT

The Mission Analysis Division of the Systems Analysis and Integration Laboratory at the Marshall Space Flight Center is developing a system of programs to handle all aspects of scheduling payload operations for Space Station. The Expert Scheduling Program (ESP2) is the heart of this system. The task of payload operations scheduling can be simply stated as positioning the payload activities in a mission so that they collect their desired data without interfering with other activities or violating mission constraints. ESP2 is an advanced version of the Experiment Scheduling Program (ESP) which was developed by the Mission Integration Branch beginning in 1979 to schedule Spacelab payload activities. The automatic scheduler in ESP2 is an expert system which embodies the rules that expert planners would use to schedule payload operations by hand. This scheduler uses depth-first searching, backtracking, and forward chaining techniques to place an activity so that constraints (such as crew, resources, and orbit opportunities) are not violated. It has an explanation facility to show why an activity was or was not scheduled at a certain time. The ESP2 user can also place the activities in the schedule manually. The program offers graphical assistance to the user and will advise when constraints are being violated. ESP2 also has an option to identify conflicts introduced into an existing schedule by changes to payload requirements, mission constraints, and orbit opportunities.

INTRODUCTION

In this paper, we shall describe the program's capabilities as seen by the user, the activity and increment constraints the program handles and how the expert system in the program handles these constraints. We have referred to activities and activity timelines, assuming that the reader has an intuitive understanding of these concepts. This discussion of ESP2 can be better understood if we first define several terms.

EXPERIMENT - In general, a collection of procedures and equipment which, when executed, contribute to the body of technological or scientific information. In ESP2, the term experiment refers to the "model" or "models" corresponding to the general definition.

FUNCTIONAL OBJECTIVE - A large section of an experiment's procedures which accomplishes a definite purpose, such as verifying equipment, collecting baseline data, etc.

MODEL (ACTIVITY MODEL) - The database representation of an experiment or part of an experiment. A model is a collection of constraint and execution definitions. Some of the definitions apply to the whole model and some apply to the "steps" of the model.

STEP - The smallest, clearly delineated part of an activity model. Steps are usually executed in sequential order, but are not necessarily contiguous. Resource and crew requirements of a model are shown at the step level.

PERFORMANCE - An execution of an activity model. A model may be performed multiple times to collect additional data.

EXPERIMENT TIMELINE - A time history of experiment performances and related activities planned for a Station increment. These activities are represented by the start and stop times of model steps.

The Expert Scheduling Program is a highly interactive, user-friendly program designed to run on a workstation with high resolution graphics, a mouse and keyboard. It checks all input for reasonableness and provides in-line help text as needed throughout the program for user assistance. The program allows the user to interrupt any of the ESP2 activities without corrupting internal or external data. A journal of all user interactions is maintained to assist in tracking the development of a schedule.

ORIGINAL PAGE IS OF POOR QUALITY

PREPARING THE INPUT DATABASE

The application of ESP2 to Space Station increment planning is a very complex process that requires several weeks to build a final timeline. An outline of the schedule building operations for a Space Station increment is shown below. This outline includes activities done by ESP2 as well as related activities done in preparation for running ESP2.

Preparation:

- * Obtain a general understanding of the increment and its objectives.
- * Develop a "gross" timeline for the flight increment. This will include such things as STS docking, equipment changeout, crew work cycle, reboost times, etc.
- * Determine the resource usage by non-scientific activities. This will include the crew off-duty periods, the sub-systems, etc.
- * Determine the names and descriptions of all observation opportunity subjects (opportunity file contents).

Modeling:

- * Obtain and enter into the increment model a description of the increment availabilities.
- * Convert or generate all subjects required to be on the opportunity file.
- * Build models of all activities (both science and non-science) that must be scheduled. These models should be checked for internal consistency.
- * Further validate these models by scheduling each one or each related group onto a new (empty) schedule. This will verify that the models can schedule.

There are several goals to keep in mind when developing activity models. The timeline engineer should always attempt to minimize the total number of models required to schedule the increment. Each model should be as simple as possible. Always minimize the number of steps on each model. Steps are required to reflect a change in any one of the resources required. Develop models that will schedule automatically.

ESP2 is the "heart" of the Experiment Scheduling System (ESS). This scheduling system will be used at several levels of Space Station planning. The Discipline Operations Centers (DOC) will use ESP2 to schedule gross models of all their users' requirements. This schedule will define the individual user's operation "windows". The user (scientist) at a User Operations Facility (UOF) will use ESP2 to build a detailed timeline for all his experiment's operation. The detailed timelines will be integrated into a coordinated Discipline activity timeline at the DOC. The Payload Operations Integration Center (POIC) will use ESP2 to integrate all of the DOC timelines. This program must be able to accommodate each of these functions. Figure 1 shows the relationship and data flow within the overall system.

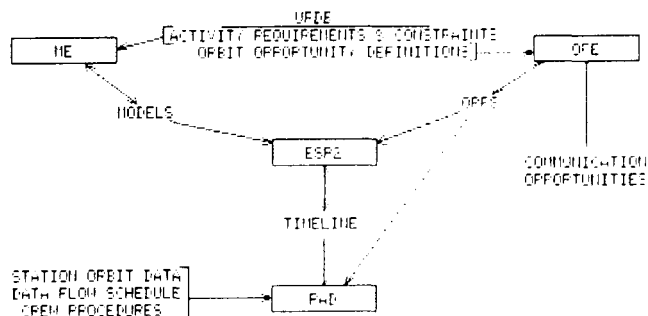


Figure 1. Space Station Timeline Analysis Flow

The scientist whose payload is scheduled to be flown on a Space Station increment will enter his experiment's requirements and constraints in the User Requirements DataBase (URDB). The Model Editor (ME) program and the Opportunity File Editor (OFE) program will use expert systems to extract this information and build files for input to ESP2. The Payload Activity Display (PAD) will use output from ESP2 to produce summary charts for the scientists, crew operations and ground support.

The Model Editor program is a database management tool used to create, modify and copy ESS activity model files used by the ESS programs. The domain of the Expert Scheduling Program includes the activity and increment constraints that are defined in the model file. These files contain the following;

- * Increment data
 - Increment start date and duration
 - Constraints and availabilities
- * Activity Models
 - Requirements and constraints
 - Grouping data
- * ESP2 Control data
 - Opportunity and external retrieval file names
 - External retrieval control data
 - Scheduling control data
 - Grading criteria

The increment data is composed of all the information required to uniquely identify a particular Space Station flight increment. This includes the name or number of the increment, start date, start time and the increment duration. Also, included is a list of the crew members. This input may also be used to specify the crew skills required for activities to be conducted during the flight increment. All resources, constraints and equipment required for the increment operations are identified and availabilities set in this data. The two types of resources are consumables and nondepletables.

Each Activity Model contains constraints and requirements for scheduling the operations that it represents. This data includes the following parameters: maximum performance duration, performance time windows, maximum number of performances, performance separation, startup/shutdown or

scenarios, sequencing, concurrence, crew lockin and resource tolerances. The experiment operations are divided into steps to specify the detailed requirements and constraints. Each model may contain up to 50 steps. The data that is specified at the step level is: duration, delays, equipment/constraint requirements, consumable requirements, nondepletable requirements, resource carry-through, fulltime crew or monitoring, and selected, intersected or avoided orbit opportunities. Each of these parameters and requirements will be explained in more detail as we later describe how they are implemented in ESP2.

In most cases, more than one Activity Model is required to represent all of an experiment's operations. The model file contains grouping data that describes which models represent each experiment and discipline.

The ESP2 control data allows the user to tailor the program's operation for specific applications. This data is saved and maintained on the model file.

SCHEDULING

ESP2 provides an array of tools for building, checking and documenting the activity timelines. There are three tools available to build the timeline. The external retriever can retrieve all or part of a previously generated timeline. The automatic scheduler can quickly add multiple performances of multiple models to a timeline. The manual scheduler allows the user to modify or delete what is already scheduled or to add new activities to the timeline. When utilizing any of these development tools, ESP2 will verify that the resources are not overbooked and the opportunity requirements are not violated in the timeline. The program can maintain, in internal storage, up to three timelines (in addition to the current timeline). ESP2 provides a complete output package that allows the user to inspect a timeline and prepare documentation. The relationship of these tools within ESP2 is shown in Figure 2.

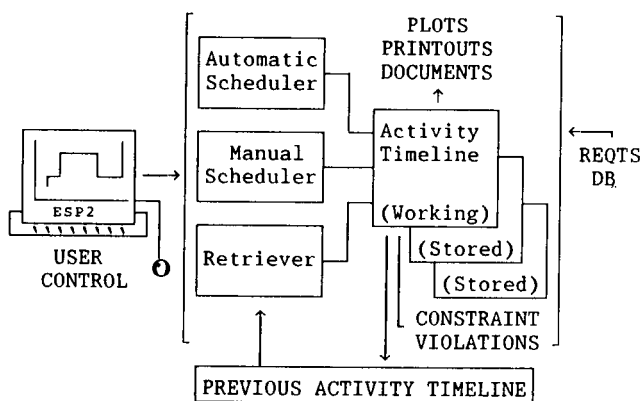


Figure 2. ESP2 Schedule Building Options

There is no hierarchy among these tools. The user may choose them in any logical order to build the final schedule. Each of these will be discussed in this section.

As a result of our experience in planning nine Spacelab missions, we have developed a prototype technique for scheduling Space Station payload operations.

- * Start by retrieving all activities that have been defined and scheduled by other sources. These normally include the crew models, sub-system models and other fixed models. A timeline file is usually written at this point so that all future runs of ESP2 can retrieve this partial schedule and add to it.
- * Schedule the high priority activities using either the automatic scheduler or the manual scheduler. This is an incremental process usually requiring several days. The activity models may be grouped according to a user defined criteria to provide a set of logical intermediate points during the development of a timeline. At the end of each terminal session, or an intermediate point, a timeline file is written. At the beginning of the next scheduling session an external retrieval from the file is performed to continue the timeline build-up.
- * Schedule low priority activities in a manner similar to the high priority activities. During this step, the resource availabilities may become so limited that many activities cannot be scheduled. The timeline developer may either change the requirements of the models or shuffle what is already scheduled to free the needed resources. The explanation facility (trace) is useful to understand why a model will not schedule or why it schedules at an unexpected time.

EXTERNAL RETRIEVAL

External retrieval is used to load into ESP2 a schedule which was previously generated and stored on a file. When using external retrieval, the user specifies the start and stop times of the retrieval and the activity models to exclude. The program does not allow double booking of a crew member or overlapping performances of the same model, but no other validation checks are made. Since the external retrieval accesses the model file, any updates to the resources (consumables, nondepletables and equipment only) will be incorporated into the timeline.

The retrieval function provides several valuable capabilities to the user. Prior scheduled activities may be retrieved, or "loaded" into ESP2, before adding more activities to the timeline. This function is necessary for complex increments that require several weeks to develop. Also, two timelines may be merged. This is done by retrieving from first one timeline file

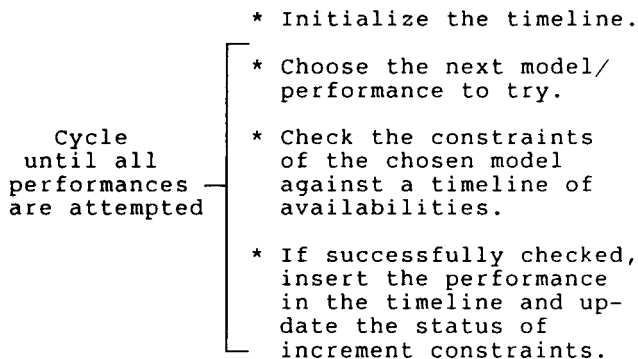
and then another. Output can be produced for a selected subset of the timeline. This is done by excluding all but the desired activity models, retrieving these models and producing the desired output.

AUTOMATIC SCHEDULER

Automatic scheduling is used to add multiple performances of multiple models to the current schedule. This option allows the program to decide where (in the schedule) to place the performances of the models. When using the automatic scheduler, the user specifies the models to be scheduled and, optionally, the order in which to attempt to schedule the models. The user may divide the models into 12 groups and specify the scheduling strategy for each group. The scheduling order does not affect the schedule except that resources are assigned on a first-come, first-serve basis. The user also specifies the weighting parameters reflecting the priority of the models, the schedule start time, the grading criteria for the schedule and the number of scheduling passes to make.

The automatic scheduler is the primary mode for developing timelines. This scheduler minimizes the time required to produce a schedule. It will build a schedule which meets all requirements. It will quickly add activities to a schedule without allowing user input errors. It can automatically generate several schedules and save the best.

A brief top-down explanation of the scheduling process in ESP2 is outlined below.



The ESP2 scheduling process is based upon calculating windows which are nested in a hierarchy such that each lower window is totally contained within the window above it. Checking of constraints begins at the topmost window and proceeds downward and across (i.e., "depth-first searching"). Checking is successfully complete whenever an acceptable window is found at the bottom level. Checking fails whenever another window at the top level cannot be defined.

Selection Methods

The choice of which model to schedule next is determined by the selection method. ESP currently provides three selection methods

for the user to choose from; Fixed, Random and Grade Maximization (^Grade). Each selection method will choose a scenario for the selected model based on upon a user specified strategy.

The Fixed method requires a user to specify the scheduling order of the models within each group. An optional command, for this method only, will order the models within each group such that the most difficult is listed first. This command computes a difficulty factor for each model based on the time windows on the model, the orbit opportunity requirements of the model, the number and duration of requested performances. These computations consider what is already in the schedule; so that different orders will be obtained by generating the order at different stages of the increment timeline development.

The Random method uses a random number seed and generator to select which model to schedule. Each performance has an equal probability of being chosen. Therefore, requesting an excess number of performances will give a model a higher probability of being selected early. This allows the user to assign a high priority to a model for the purpose of scheduling. By choosing the random selection method and requesting many scheduling passes (trials), the user can effect a Monte-Carlo approach to finding the best solution.

The ^Grade method evaluates which model will provide the greatest improvement in the schedule grade and selects it to be scheduled next. With this method, ESP2 dynamically updates the selection order as each model is scheduled. The grading criteria selected by the user will affect this selection. The grade is based upon the following five factors: the number of performances scheduled, the number of activity models scheduled, the amount of crew time utilized, the amount of activity operation time scheduled and the mean number of performances scheduled.

Loading Algorithms

The placement of the model in the timeline is determined by the loading algorithm and when the requirements of the model are met. This algorithm controls where each performance is scheduled if it is not precisely fixed by performance time windows, performance separation times or other model constraints. ESP2 currently provides two loading algorithms; Front and Back. Front loading will always attempt to schedule the performance at the earliest time which satisfies all constraints. Back loading will always attempt to schedule the performance at the latest time which satisfies all constraints. The loading algorithm can be selected independently for each group of models to be scheduled.

The activity requirements and constraints defined in the model file are not rule based. Each activity model has several

different types of requirements and constraints which determine the time at which the model may be scheduled. ESP2 has a fixed rule base which handles all of these requirements. These requirements may be classified as: time constraints, performance options, relational constraint, orbit opportunities and resources.

Time Constraints

An activity model may require that all performances be scheduled within specified time windows. Each model may have up to 10 windows with a start time, a stop time and the number of performances required for each. Multiple performance windows allow the timeline engineer to predictably and variably space performances throughout the schedule.

The size of the scheduling window for a model can be constrained by limiting the maximum performance duration. This gives the user control over the length of an entire performance while allowing ESP2 to stretch out steps as needed. An activity model may also limit the separation between adjacent performances. ESP2 will always attempt to minimize the performance separations.

Each step of an activity model contains limits for the duration and the time to delay before beginning step operations. These limits are defined in the model database with a minimum and maximum value. ESP2 will attempt to maximize step duration (as long as all other requirements are fulfilled) and minimize step delays.

Performance Options

There are two mutually exclusive options for defining the steps to be scheduled on each performance. These options are defined in the models during the ME build process. ESP2 is capable of scheduling either option without any further user interaction.

The first option is Startup/Shutdown. A model may have startup steps which are executed only for the first performance of an activity and shutdown steps which are executed only for the last performance of an activity. However, there must be a core of steps (at least one) which are executed on all performances. With this capability, ESP2 can automatically schedule activity startup/shutdown with a single model.

The second option is Scenarios. A scenario is an alternate ordering of the steps of an activity model. A model may define up to four scenarios. Each scenario has a priority associated with it. Currently, ESP2 has a limited rules set for handling scenarios. This is an area that is still being developed and could provide major improvements in the scheduling capabilities.

Relational Constraints

ESP2 also provides techniques for scheduling experiment activities relative to other Space Station activities. These techniques are sequencing and concurrence. Sequencing is accomplished by requiring a model to be scheduled within a specified time period after the performances of a leading activity model. This time window is restricted by a minimum and maximum value. ESP2 will attempt to minimize the sequence delay. By manipulating these parameters and the performance separation, the user can implement this as either a prerequisite or a one-to-one sequencing requirement. Concurrence is accomplished by requiring that one step of a model be scheduled at the same time as a step of another model. ESP2 provides three levels of concurrence: Mandatory, Necessary and Desired. Each level is handled differently by the scheduler.

Orbit Opportunities

An activity model step may specify a set of orbit opportunities that must exist during its operations. An orbit opportunity may be defined as (1) a celestial object or ground site which is to be observed or (2) an Experiment observation window. Each model step may have up to three categories of requirements for the opportunities. These are Intersected, Selected and Avoided. ESP2 will process any subject on the opportunity file as one of these requirements. Up to three subjects may be intersected for a specific step. The step will be scheduled only if all three subjects exist for the entire duration of the step. A list of up to five subjects may be processed as selected opportunities. The step will be scheduled if at least one of the subjects is available for the entire duration of the step. The list of 5 selected opportunities may also be avoided. In this mode, the step can only be scheduled when none of the opportunities are available for any part of the activity.

Resources

A resource used in discrete integral amounts, such as a TV camera, whose availability is temporarily changed for the duration of its usage is called an Equipment/Constraint. Each activity model step may require up to 15 different types of Equipment/Constraints. In many cases this type of resource is used to prevent two models from scheduling at the same time. It can also be used to make models schedule concurrently and for "bean counting".

The availability of a Nondepletable resource is also temporarily changed only for the duration of its usage. However, this type of resource may be used in fractional amounts (e.g., Power). An activity model step may require up to 10 nondepletables. A negative usage rate will increase the availability for the duration of the step.

A Consumable resource is one whose availability is permanently changed by its usage (e.g., Camera film). Each model step may require up to 10 different types of consumables. A consumable can be defined to be used at a fixed amount, constant rate or based upon a nondepletable rate.

An activity model step may require resource Carry-Through. This function, when used in conjunction with step delays, allows ESP2 to schedule resource usage during a delay between activities.

The Payload Crew is possibly the most valuable resource for any Spacelab mission or Space Station increment. The scheduling of crew members to perform the experiment activities is given special consideration. ESP2 will permit the user to override warnings about overuse of other resources, but it is impossible to overbook a crew member (i.e., schedule him/her to perform more than one activity at once). The program also provides for fulltime crew participation or periodic monitoring. Each model step may specify up to three lists of the eight possible crew members. These lists can be used to identify the crew according to their skill levels. The total number of crew members required for a step can be distributed across the skill levels, but cannot exceed eight. ESP2 allows crew lockin (i.e., require that the same crew members be scheduled for all steps).

MANUAL SCHEDULER

The manual scheduler, also known as the timeline editor, is used to modify existing activities or to enter activities whose requirements are not well modeled or whose placement is predetermined. When using the timeline editor, the user enters the start and stop times and crew usage for each step of an activity. Resource usage is taken from the model steps. The editor presents a screen of data to be edited using form-editing techniques and commands. When the user issues the command to commit the page to the schedule, the entries are checked for constraint violations. If none of the violations will not destroy the integrity of the schedule, the user may ignore the warnings and update the schedule. A chart showing where each required resource of an activity is available and the intersection of these availabilities is provided to assist the user.

The editor also provides easy access to the automatic scheduler for quick scheduling of a model (possibly with some requirements overridden). This override feature will temporarily change a limited set of model requirements within ESP2 only. The model database is not changed.

CHECKPOINTING

Checkpointing/restarting is used to save a schedule internally; and, at some later time, restart from that checkpoint. ESP2

has four slots which may contain schedules. Each checkpoint is timetagged and labeled to identify its contents. One of these slots is used to maintain the current or working schedule. All changes are made and all output is generated from this schedule. The second slot used by the program maintains a copy of the last external retrieval. Two additional checkpoints are available for the user. During the development of a timeline, it is good practice to periodically create a checkpoint to save the existing schedule.

EXPLANATION FACILITY

The engineering trace is an explanation facility which can show the user why ESP2 could not schedule another performance of a model or why ESP2 scheduled the model where it did. ESP2 cannot tell why a model cannot be scheduled. Indeed, there is often not a single reason. The crew may be available when the orbit opportunity is not, or the orbit opportunity may be available when the power is not. The reason for failure was not the lack of crew or the lack of power or the lack of an orbit opportunity. The reason can be stated: "because all requirements were not met at the correct times". ESP2 provides a graphic and textual trace of the checking and loading portions of its scheduling algorithms so that a proficient user can understand how ESP2 arrived at a schedule and therefore why something is not scheduled or is scheduled at a particular time.

OUTPUT

ESP2 documents experiment timelines! The array of output generated upon user request includes tabulations, terminal and laser plots, document outputs, printouts, and experiment timeline and on/off files. A complete list of the types of output available in the current program is shown in Figure 3.

Most of these output options utilize a control form. This form presents a choice of up to six output mediums to accommodate the various hardware capabilities. When ESP2 is run on a non-graphics terminal, the terminal plot options are not available. However, the plots can be generated on a Laser printer. When ESP2 is run on a workstation, the program is capable of displaying up to 9 overlaid terminal plots. These plots can be reduced to icons for later recall.

One of the most useful output formats is the Step Opportunity Plot. This output shows all the requirements of a step and where these requirements are met. The intersection of the time periods where all requirements are met is plotted as the opportunities to schedule the step. This is a valuable tool when building up a timeline.

1- Composite TL (Timeline file)	TPF
2- Subset Composite TL	TP
3- Selected Activity TL's	T F
4- Crew Timelines	TPF
5- Selected Equipment TL's	TP
6- Equipment/Constraint Usage	TPF
7- Selected Nondepletable TL's	T
8- Nondepletable resource usage	TP
9- Schedule Overview	T
10- Subset Composite Overview	T
11- Activity Overviews	T
12- Performance Summary	T
13- Crew Work Summary	T
14- Crew Utilization Summary	T
15- Crew Availability Summary	T
16- Unscheduled Performances	T
17- Activity Summary	T
18- Average Nondepletable Usage	T
19- Nondepletable Minimum Avail.	T
20- Step Opportunities	TP
21- Performance Opportunities	TP
22- Models	T
23- Orbit Opportunities	TPF

AVAILABILITIES:

T - Tabulation or Printout
P - Plots
F - File

Figure 3. ESP Output Menu Options

CONCLUSIONS

Although the ESP2 program is still being developed, its predecessor, ESP, has been thoroughly qualified. ESP has successfully supported the activity timeline development for nine Spacelab missions, three of which have flown, and several partial payloads. This program is currently being used not only by MSFC, but also by the Flight Project Engineering Office at JSC to schedule the SLS-1 and SLS-2 Spacelab missions, Martin Marietta Denver Aerospace to schedule the TSS mission and Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), Germany's equivalent to NASA, to schedule the D-2 Spacelab mission. Langley Research Center recently requested a copy of ESP for evaluation of its applicability for their Space Station evolution studies.

ESP is being used at MSFC today to schedule 90-day strawman flight increments for Space Station investigations. This exercise will provide more insight into the modifications required for ESP2 to handle the Space Station scheduling task. ESP2 will utilize AI technology, where it is cost effective, to enhance the program capabilities. Some currently planned improvements include; the ability to edit the activity models within ESP2, the ability to generate observation opportunities within ESP2, and additional loading algorithms. The capacities of the program, as it was designed for Spacelab operations, and the maximum usage experienced for the three successfully completed Spacelab missions are listed in Figure 4.

PROGRAM CAPACITY	HISTORICAL PEAKS		
Mission duration (days)	1000	10	(SL-1)
Activity models	500	361	(SL-1)
Performances per Model	500	135	(SL-1)
Performances per Timeline	50000	1218	(SL-1)
Steps per Model	50	37	(SL-1)
Steps per Timeline	50000	5163	(SL-2)
Crew members	8	7	(SL-3)
Equipment/Constraint types	99	67	(SL-1)
Nondepletable resources	25	5	(SL-3)
Consumable resources	25	7	(SL-1)
Opportunity subjects	200	109	(SL-2)
Acq/Losses per opportunity	500		

Figure 4. ESP Program Capacities

During the Spacelab era, the average overall usage of these capacities was only about 10 percent. Therefore, we believe these limits will be a good starting point for scheduling Space Station payload operations.

In a recent interview for the Marshall Star, John Jaap summed up the status of ESP2 development thus; "Scheduling for Spacelab missions is complex enough, even for short missions. On the other hand, Space Station activity scheduling, with more shared resources and experiment durations counted in weeks rather than hours, requires a more automated computer program. For Space Station we're adding features to ESP that will automate many of the tasks that now must be done by an expert user. New rules and strategies for scheduling are being developed."

REFERENCES

1. Dumoulin, John. "ESP Keeping Spacelab Experiments Conflict-Free," Marshall Star, vol. 28, number 33 (April 27, 1988): page 2.
2. Jaap, John P. and Davis, Elizabeth, "Experiment Scheduling for Spacelab Missions", Conference on Artificial Intelligence for Space Applications, Huntsville, Alabama, November 13-14, 1986.
3. Jaap, John P. and Davis, Elizabeth, "The Scheduling Techniques of ESP2", Second Annual Workshop on Space Automation and Robotics, Dayton, Ohio, July 20-23, 1988.
4. Jaap, John P. and Davis, Elizabeth, Experiment Scheduling Program User's Manual. Huntsville, Alabama: Marshall Space Flight Center, [1988].
5. Maxwell, Theresa G. "Investigation of Experiment Selection Order Determination for Space Mission Scheduling." M.S. thesis, University of Alabama in Huntsville, 1987.
6. Stacy, Kenneth Lamar., Training Course 5200: Mission Activity Model Development, Huntsville, Alabama: Marshall Space Flight Center, [1988].
7. Stacy, Kenneth Lamar., Training Course 5420: Timeline Editing. Huntsville, Alabama: Marshall Space Flight Center, [1988].

A Survey of Planning and Scheduling Research at the NASA Ames Research Center

Monte Zweben
NASA Ames Research Center
Moffett Field, CA 94035
(415) 694-6940
zweben@pluto.arc.nasa.gov

Abstract

NASA Ames Research Center has a diverse program in planning and scheduling. This paper highlights some of our research projects as well as some of our applications. Topics addressed include machine learning techniques, action representations and constraint-based scheduling systems. The applications discussed are planetary rovers, Hubble Space Telescope scheduling, and Pioneer Venus orbit scheduling.

1 Introduction

NASA Ames Research Center's Artificial Intelligence Research Branch, led by Dr. Peter Friedland, has a diverse research program in planning and scheduling. Our work ranges from state-of-art fundamental research to applications of both new and existing technology. This paper is intended to summarize and highlight some of these activities.

The research issues we will highlight include: machine learning and planning, planning representations, non-symbolic representations, constraint-based scheduling, and the representation of procedural knowledge.

The applications we will present include Hubble Space Telescope scheduling, Mars Rover planning and scheduling, and Pioneer Venus orbit scheduling.

2 Planning and Scheduling

It is important to clarify the terms "planning" and "scheduling" before we proceed. An agent *plans* by finding actions that will take it from its current state

to another desired state. Classically, this is a goal directed search through a space of possible partial plans. *Scheduling*, on the other hand, refers to an agent placing explicit times or orderings on a set of intended actions. This is usually a search through a space of possible timelines. In short, we call the process of finding actions that achieve goals *planning* and we call the placement of times on those actions *scheduling*.

3 Research

Our research program is a mix of internal research, university grants, and commercial contracts. Here we will present a representative subset of the program conducted at Ames, SRI, Stanford, and Carnegie-Mellon.

3.1 Learning in Planning

One of our group's areas of focus is machine learning and we are particularly interested in its application to planning and scheduling. We are exploring ways to improve search performance through the application of explanation-based learning techniques [Mit87, DeJ87]. The main idea behind this work is that a system can improve its performance by analyzing the solutions to problems it has previously encountered. As a result of this analysis, the system can remember the good decisions it made as well as the poor ones. Ideally, we would like the system to generalize from this analysis so that the knowledge gained from its retrospection will be useful in cases that are not only identical to the ones it encountered, but also those that are close enough so that the previous experience would prove relevant and helpful.

PRECEDING PAGE BLANK NOT FILMED

Dr. Steven Minton, of Carnegie-Mellon University, performed a thorough analysis of a planning and learning system called PRODIGY [Min87,Min88]. PRODIGY is a STRIPS-like planner that employs explanation-based learning to acquire search control knowledge. His results showed that learning will not necessarily improve the performance of a planning system and in many cases it can degrade performance. As a result, Dr. Minton explored various methods of monitoring the utility of learned knowledge in order to transform (or possibly remove) learned knowledge to make the overall system more useful. Dr. Minton has recently joined our laboratory and will continue exploring planning and learning issues.

Another project within our laboratory is also addressing the utility problem in planning systems that learn. Monte Zweben and collaborators at the MITRE Corporation are specifically addressing the utility problem caused by the complexity of learned knowledge [Zwe88b]. When a planning system needs to make a decision it must consider the generalized information that it has learned. This pattern-matching overhead can overwhelm the system to the point where learned knowledge no longer aids efficiency. Using PRODIGY as a model, Mr. Zweben and his colleagues are developing a system that employs explanation-based learning (EBL) to acquire search knowledge, but relaxes some of the constraints usually associated with EBL techniques. Specifically, EBL generalizes from a single instance and guarantees the correctness of the learned knowledge. As a result, the learned information tends to be quite complex. This project's main extension to the PRODIGY model is the approximation of learned knowledge in the interest of lowering the expense of the relevancy check. As a result, this approximation of learned knowledge could be incorrect and must be monitored. If the learned knowledge is approximated erroneously and misleads the planner frequently, then the approximations must be refined. The goal of this project is to determine the approximation and refinement strategies that will result in an efficient and effective collection of knowledge learned by an explanation-based component.

3.2 Planning Representations

Dr. Mark Drummond, of our group, takes a Net Theory approach to the problem of planning, scheduling and control [Dru85,Dru87]. His approach has a number of interesting features and advantages. Similar to Amy Lansky's [Lan87] work, it views a plan as a set

of constraints over a pre-specified set of actions. Unlike Lansky's GEM model, however, the Net Theory approach allows one to distinguish clearly between orderings required by causality, and those that are simply convenient, given the agent's goals. The Net Theory approach also begins to make clear the true role of *least commitment planning*, where orderings on actions are postponed until an ordering decision *must* be made. Current plan representations frequently over-commit to specific orderings. This over-commitment is critical when dealing with complicated scheduling problems, since many orderings and conditions cannot be determined until a schedule is actually being carried out. The Net Theory approach currently being explored by Dr. Drummond allows complete postponement of ordering decisions until all environmentally determined information is available. This permits a new view on the role of an agent's synthetic temporal data structure. These data structures can now be viewed as plans, schedules, or control programs, depending on the phase of overall system operation. This work does not view planning and scheduling as a one-time process, but rather, includes an explicit control phase where plans/schedules are incrementally modified to suit execution needs.

Dr. Drummond is also exploring a number of other issues in his planning research including: the tradeoff of reactive and predictive scheduling, the role of means-ends analysis in planning, the integration of planning and scheduling mechanisms, the representation and derivation of conditional and iterative plans, the role of constraint-satisfaction in the planning process, and the use of domain constraints to control planning search [Dru88].

3.3 Control Without Symbols

The work of Dr. Stan Rosenchein, formerly of SRI International and now of Teleos Research, takes the perspective that expensive symbolic processing at run time can be avoided by compiling symbolic representations into circuitry guaranteed to act in bounded time. Dr. Rosenchein and his colleague Leslie Kaelbling have developed a set of tools that enables one to design a robotic controller in a high-level language, which then gets compiled into efficient circuitry that can be simulated or manufactured in hardware [Kae88,Ros86]. The fundamental idea behind this work is that much of the expensive search (like pattern matching) employed by symbolic reasoners can be accomplished at compile time, allowing the robot to quickly process its sen-

ORIGINAL PAGE IS OF POOR QUALITY

sory information and react appropriately. One of their tools, Gapps [Kae88], takes a goal expression and rules in a goal decomposition language and outputs circuitry that will enable a system to take action given a goal and its current state. Their tool REX allows one to specify behavior that takes sensory input and the system's current state and updates the current state to reflect what has occurred in the system's environment. REX allows one to specify the circuitry in a language more abstract than circuits, but less abstract than that of a programming language. They are currently designing a system called RULER which will allow one to design the state update circuitry in a logical language resembling PROLOG. Ultimately, this language will be compiled into REX specifications.

This work is distinguished in that the REX language has been specifically designed to support analysis of any particular REX program to prove its correctness. Further, this work is currently used to control Flakey, the SRI mobile robot. We view this work as a realistic first step towards the production of efficient robotic control tools. It begins to show how a designer can allocate computational resources at different phases of the design and deployment process.

3.4 Constraint-based Scheduling

As previously mentioned, scheduling is the process of placing a pre-specified set of actions on a timeline ensuring that the schedule's constraints are maintained. One of our projects, led by Monte Zweben, addresses the formulation and resolution of complex scheduling and resource allocation problems using constraints to represent scheduling knowledge and preferences [Zwe88a]. Constraints are declarative representations of relationships that abstract away control flow. They allow one to specify the relationships between the problem's variables in a system and enable the system to automatically determine the computation path from known variables to the unknown [Sta77]. These representations can be used for lookahead in a search process. Lookahead or constraint propagation results in less backtracking (i.e., fewer futile search paths) because commitments to various choices in the system are made only if they are compatible with the choices remaining in the system [Har80, Ste80]. However, lookahead can result in unnecessary constraint propagation. To circumvent this problem, we employ a technique called delayed evaluation [Fil84]. A system employing delayed evaluation does not completely evaluate its data structures until they are accessed. We

use the data structure *streams* [Abe85] which are lists that delay the evaluation of their tails (i.e., all the elements of the list except the first element). The use of streams is advantageous for two main reasons: 1) their delayed evaluation circumvents unnecessary constraint propagation; 2) their delayed evaluation is transparent to knowledge engineers because stream operations are quite similar to list operations and our model of constraint-satisfaction is based upon list operations.

3.5 Procedural Knowledge

Dr. Michael Georgeff of SRI International has developed a system called PRS - Procedural Reasoning System - that enables one to represent and use complex procedural knowledge [Geo86]. PRS takes a set of procedures and executes them in a goal-directed manner. It uses a declarative representation of procedures that extends the expressiveness of previous action representations. Actions in PRS can exhibit iteration and recursion and also can employ run-time conditional branching. Thus, decisions as to what action to perform next can be dependent upon the runtime environment. PRS procedures can also be interrupted by other procedures, thereby allowing emergency recognition and exception handling. The ability to change its focus of attention quickly and to act conditionally makes PRS a highly reactive system.

PRS also has interesting theoretical aspects in that it meets much of the rational agency criteria proposed in the recent philosophical literature. Because PRS behaves like a rational agent there is potential for the development of interesting explanation components. PRS has been exercised in a very complex and interesting domain: malfunction handling for the reaction control system of the Space Shuttle. NASA diagnostic manuals were encoded in PRS resulting in an extensible set of semi-autonomous procedures.

4 Applications

The Ames AI Research Lab performs state-of-the-art research, but does so in the context of real-world applications. This allows us to both verify that our methods scale-up to real problems and focus our research towards topics of interest to NASA. In addition to framing our research within NASA problems, we also demonstrate the utility of known AI techniques with engineering applications. Don Rosenthal is the director of our applications work. His applications projects include Pioneer Venus satellite scheduling and Hubble

Space Telescope scheduling. In fiscal year 1989, Mr. Rosenthal will explore planetary rover applications.

4.1 Pioneer Venus

This project, now completed, showed the utility of rule-based systems for operational software [Ros88]. We developed a heuristic ground-based scheduler for science operations (e.g., instrument configurations, data storage and playback, telemetry, etc.) onboard the Pioneer Venus satellite. This software is currently performing a task in minutes which formerly took people hours. Further, the resulting schedules are as effective as the man-made ones but contain fewer flaws. The satellite's operations are currently scheduled with this expert system. This scheduler is the first expert system installed in day to day use within a NASA mission operations environment.

4.2 Hubble Space Telescope (HST) Scheduling

Thousands of proposed observations for HST must be processed by the Space Telescope Science Institute (STScI), on the Johns Hopkins University campus in Baltimore, to construct schedules for the science operations of the orbiting optical observatory. Current software is not flexible or extensible enough to meet the operational demands expected on the system and we are helping to provide knowledge-based solutions to this problem [Mil87].

The HST projects we support take a constraint-based approach to scheduling. Dr. Stephen Smith, of Carnegie-Mellon University, is applying research in factory scheduling [Fox83, Smi86] to the HST problem. This approach is well suited for over-constrained problems where a solution requires the relaxation of constraints.

Another project, at the STScI, is applying state-of-the-art constraint satisfaction techniques to the HST scheduling problem. Their goal is to produce a flexible and extensible scheduler that can dynamically react to anomalies and re-schedule accordingly. This work has resulted in a program called SPIKE, which uses piecewise constant functions to quantitatively represent the degree of constraint violation. Using these functions, SPIKE can efficiently combine constraints as well as judge the options it must choose.

4.3 Planetary Rovers

In the coming year we will begin performing extensive research into the planetary rover problem while concentrating on the science planning and scheduling issues. Using the Mars Rover domain as a model, we are interested in rovers that can autonomously plan and execute an appropriate set of scientific analyses for many different science goals. Further, we will explore techniques that dynamically discover interesting science opportunities, and attempt to replan the rover's actions to accommodate these new goals.

Additionally, we will address the integration of navigation planning and science planning which will require research in systems that negotiate for resources and time.

We will also explore machine learning techniques that can improve the overall rover system. First, we will explore techniques that improve a system's search performance. Second, we will address model refinement for rovers that begin with a rough and incomplete model of their environment. These techniques review a system's actions and remembers when they succeed and when they fail. They also find discrepancies between a system's expectations and its observations and uses these discrepancies to refine the system's models.

5 Summary

This paper is intended to selectively introduce our research and to point out references to technical papers. Some of the areas currently addressed by our group but not discussed here are: 1) planning with incomplete models [Car87b, Car87a], 2) the use of truth-maintenance in planning [Mor86], and 3) communicating, cooperating agents [Nil87]. In the coming year, we plan to expand our efforts in multi-agent planning and constraint satisfaction. The overall goal of the program is to develop the technology for large-scale automation of space missions.

6 Acknowledgements

Thanks to Mark Drummond, Peter Friedland, Steve Minton, Don Rosenthal, and Haym Hirsh for their contributions to this survey.

References

- [Abe85] Abelson, H., and Sussman, G.J. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, 1985.
- [Car87a] Carbonell, J.C. and Gil, Y. Learning by Experimentation. In *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Car87b] Carbonell, J.G., Mason, Matthew T., and Mitchell, Tom M. Machine Learning and Planning in Reactive Environments. In *NASA Ames AI Forum Proceedings*, 1987.
- [DeJ87] DeJong, G.F. and Mooney, R. Explanation-Based Generalization: An Alternative View. *Machine Learning*, 1(2), 1987.
- [Dru85] Drummond, M.E. Refining and extending the procedural net. In *IJCAI-9 Proceedings*, Morgan Kaufman, Palo Alto, CA, 1985.
- [Dru87] Drummond, Mark E. A Representation of Action and Belief for Automatic Planning Systems. In *Proceedings of the AAAI/CSLI 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Dru88] Drummond, M.E., and Currie, K.W. Exploiting temporal coherence in non-linear plan construction. *Computational Intelligence Journal*, to appear, 1988.
- [Fil84] Filman, R.E., Friedman, D.P. *Coordinated Computing: Tools and Techniques for Distributed Software*. McGraw-Hill Book Company, New York, NY, 1984.
- [Fox83] Fox, Mark S. *Constraint-Directed Search: A Case Study of Job Shop Scheduling*. PhD thesis, Carnegie-Mellon University, 1983.
- [Geo86] Georgeff, M.P. and Lansky, A.L. Procedural Knowledge. *Proceedings of the IEEE*, Special Issue on Knowledge Representation, 1986.
- [Har80] Haralick, R.M., Elliot, G.L. Increasing Tree Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14, 1980.
- [Kae88] Kaelbling, L.P. Goals As Parallel Program Specifications. In *AAAI-88*, Morgan Kaufman, Palo Alto, CA, 1988.
- [Lan87] Lansky, Amy L. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Mil87] Miller, G., Rosenthal, D., Cohen, W., and Johnston, M. Expert Systems Tools for Hubble Space Telescope Observation Scheduling. *Telematics and Informatics*, 4(4), 1987.
- [Min87] Minton, S. and Carbonell, J.G. Strategies for Learning Search Control Rules: An Explanation-based Approach. In *IJCAI-87 Proceedings*, Morgan Kaufman, Palo Alto, CA, 1987.
- [Min88] Minton, S. Empirical Results Concerning the Utility of Explanation-based Learning. In *AAAI-88 Proceedings*, 1988.
- [Mit87] Mitchell, T.M., Keller, R.M., and Kedar-Cabelli, S.T. Explanation-Based Generalization: A unifying view. *Machine Learning*, 1(1), 1987.
- [Mor86] Morris, P.H. and Nado, B. Representing Actions with an Assumption-based Truth-Maintenance System. In *AAAI-86*, 1986.
- [Nil87] Nilsson, Nils J., Cohen, Phillip R., Rosenchein, Stanley J., and Fertig, Kenneth. Intelligent Communicating Agents. In *NASA Ames AI Forum Proceedings*, 1987.
- [Ros86] Rosenchein, S.J. and Kaelbling, L.P. The Synthesis of Digital Machines with Provable Epistemic Properties. In *Proceedings of the Conference on Theoretical Aspects of Reasoning and Knowledge*, Morgan Kaufman, Palo Alto, CA, 1986.
- [Ros88] Rosenthal, D.A. and Jackson, Robert W. Development of a Pioneer Venus Expert Scheduling System. In *Proceedings of the 26th Aerospace Sciences Meeting AIAA-88*, 1988.
- [Smi86] Smith, S.F., Fox, M.S., and Ow, P.S. Constructing and Maintaining Detailed Production Plans: Investigations into the development of Knowledge-based Factory Scheduling Systems. *AI Magazine*, 7(4), 1986.

- [Sta77] Stallman, R.M., Sussman, G.J. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9, 1977.
- [Ste80] Stefik, Mark. *Planning With Constraints*. PhD thesis, Stanford University, January 1980.
- [Zwe88a] Zweben, M., Eskey, M., and Rosenthal, D. Constraint Satisfaction with Delayed Evaluation. *Submitted to CACM*, 1988.
- [Zwe88b] Zweben, Monte and Chase, Melissa P. Improving Operationality with Approximate Heuristics. In *Proceedings of the 1988 Spring Symposia Series in Explanation-based Learning*, 1988.

AN EXPERT SYSTEM FOR SHUTTLE
AND
SATELLITE RADAR TRACKER SCHEDULING

Paul Mitchell

Artificial Intelligence Section - FM72
Technology Development and Applications Branch
Mission Planning and Analysis Division
NASA/Johnson Space Center
Houston, Texas 77058

Abstract:

This expert system automates and optimizes radar tracker selection for shuttle missions. The expert system is written in the Fortran and C languages on an HP9000. It is portable to any UNIX machine having both ANSI-77 Fortran and C language compilers. It is a rule based expert system that selects tracking stations from the S-band and C-band radar stations and the TDRSS east and TDRSS west satellites under a variety of conditions. The expert system was prototyped on the Symbolics in the ART and ZetaLisp. After the prototype demonstrated an acceptable automation of the process of selecting tracking stations to support the orbit determination requirements for Shuttle missions, the basic ART rules of the prototype were ported to the HP9000 computer using the CLIPS language. CLIPS is a forward-chaining rule-based expert system language written in "C". Prior to the development of this expert system the selection process was a tedious manual process and, expensive in terms of human resources. Manual tracking station selection required from 1 to 2 man weeks per mission; whereas the expert system can complete the selection process in about 2 hours.

Introduction:

Expert system technology is a major subset of Artificial Intelligence (AI) and has been aggressively pursued by AI researchers since the early 1970's. In the last few years, both government and commercial application developers have given expert systems considerable attention as well. Expert systems have a number of characteristics which make them an attractive solution to problems which require complex expertise: they don't quit, retire, or become ill; they always work at their highest level of capability; they can be applied to hostile environments or repetitious problems; and they potentially offer a computer-based solution to problems that previously required a human expert.

This paper describes an expert system that automates the laborious and time-consuming process of selecting radar tracking systems prior to each Shuttle flight. The tracker selection is necessary in order to insure that the mission has adequate tracking coverage to provide required position and velocity information. The data obtained from these trackers provides the information for the orbit determination process in the MCC and therefore is crucial for performing maneuvers.

This development expert system also served as a case history for the development and delivery of expert systems. The initial prototype of TRACKEX was developed in ART on a Symbolics machine in three weeks. There were approximately one hundred ART rules with an average length of twenty lines. There were also five hundred lines of LISP code, primarily providing a user interface. The target machine for the delivery system was an HP9000. The conversion of the one hundred ART

rules to CLIPS took one week, and three weeks were spent in converting the LISP to FORTRAN.

EXPERT SYSTEM OPERATION:

The expert system TRACKEX is a rule-based software package written in the C and FORTRAN languages. The rules are coded in the production rule language CLIPS. The use of CLIPS was driven by the choice of the target machine: a unix-based HP9000, inputting the C and FORTRAN compiler. FORTRAN subroutines and C functions were developed to replace the LISP I/O interface of the prototype. The released product consists of 295 rules, 52 FORTRAN subroutines and 23 user-defined C functions.

TRACKEX assigns the best possible group of tracking systems from the total set of trackers available during each orbit. This selection is made with consideration given to geometry, no two stations from the same geometric area; to expense of operation, some sites are always active and are therefore more economical to use than a site that has to be manned specially for the mission; and to subsequent orbit coverage, especially for a Target vehicle. Once a tracker is assigned to a Target, it is assigned in subsequent orbits until its elevation angle, with respect to the tracker's local horizontal, falls below an acceptable limit.

TRACKEX will make tracker selections for up to four vehicles for batches of ten orbits at a time. The user defines the preferred priorities when more than one vehicle is to be considered and this priority ordering is maintained throughout the entire selection process unless the user modifies the priorities during the run. TDRSS and S-band systems can track multiple vehicles during the same orbit. However, most Targets will not have S-band transponders and will therefore be skin tracked. Skin tracking prohibits the tracking of more than one vehicle at a time. Therefore, multiple Targets may mean sparsity of available trackers, making the assignment of vehicle priority very important.

The number of user inputs are minimal and are menu driven thus, training a user to use the system is quickly accomplished. The expert system can be run in a completely automatic mode after the program has been properly initialized and all user required inputs are made.

The tracker selection matrix table is described in the appendix, and the included table is an example of the expert system's output using simulated data. The table is formatted according to the users' specifications and is intended to be adequate for insertion into all documentation requiring it's information content.

TRACKEX has proven to be both accurate and fast. During the verification phase, it was found to be correct in these cases where the initial human expert solution differed from the expert systems selection.

The verification for TRACKEX was an exhaustive case-by-case comparison of human expertise versus TRACKEX automation. This was possible since there are only four groups of ground stations, each containing a limited number of specific tracking sites. The verification cases were designed so that each tracker would be selected at least once at the appropriate time and under a variety of conditions. Each of the three tracking phases, non-critical, critical, and complex were exercised to the fullest extent.

The human expert designed the test cases to cause every rule in the expert system to be fired at least once. The expert then made independent tracking system selections per case prior to running TRACKEX. When there was a difference in the selections, TRACKEX was ultimately found to be correct.

In addition to being accurate, TRACKEX is capable of making the tracker selection for an entire Shuttle Mission in about two hours of computer time, whereas the manual method takes from one to two man weeks and is subject to error (a less than optimal selection) because it is very difficult for a person to remember and constantly apply 295 rules.

Conclusion:

Although an entire industry has grown to support the development of expert system tools and applications, with a wide variety of both hardware and software products now available, expert systems have generally failed to make a major impact in application environments where there is a requirement for specific hardware or integration with procedural languages such as ADA, C, or FORTRAN. The problem of delivering a functional application, especially in embedded systems, has proven to be a major stumbling block.

This stumbling block was avoided in this project by using the CLIPS language for the delivery of TRACKEX.

The CLIPS language provides expert system technology in a conventional language with the capability of complete integration with conventional procedural languages.

Acknowledgments:

The author wishes to acknowledge the most appreciated aid provided by Capt. Richard Kidd, USAF for his expertise in the selection criteria and to Chris Culbert, Gary Riley, and Lui Wang for their technical assistance.

References:

Inference Corporation, ART Reference Manual, (Los Angeles, California: 1986).

Culbert, Chris, CLIPS Reference Manual, NASA Technical Memo FM8(87-220), (Houston, Texas: NASA/JSC, 1986).

Giarratano, Joe, The CLIPS User's Guide, NASA Internal Note 86-FM-25, (Houston, Texas: NASA/JSC, 1986).

Mitchell, Paul, Completion and Delivery of TRACKEX, NASA Technical Memo, FM7(87-353), (Houston, Texas: NASA/JSC, 1987).

Appendix:

DESCRIPTION OF THE TRACKER SELECTION MATRIX

The output tables containing the tracker schedule are stored in permanent files for inclusion in documentation requesting the trackers to be scheduled for a particular orbital mission.

The following is a description of each entry to the table:

- a) Title---Tracker Support Matrix
The table is a matrix of the selected tracker site (row) vs the orbit in which it is available (column).
- b) Sub-titles
Flight: The flight acronym is picked up automatically from the visibility files which contain the flight's predicted tracker acquisition and loss of signal profile.

Orbits: The ten (10) orbits over which the current table extends are indicated by their starting orbit number and the end orbit number for quick reference purposes.

Cycle: This is an integer that indicates the version number for documentation purposes.

The full date and time of the table's creation is included again, for documentation and referencing purposes.

- c) The table entries
The column titled "SITE" contains the Tracker acronym for a Tracker (S-band, C-band, or Tdrss) selected at least once during the ten-orbit period. For instance the first acronym is TDRE and it has entries in all ten orbits.

The entry itself is coded such that the following information is revealed:

- a) The first single symbol (C, X, or *) indicates the vehicular phase for that particular vehicle. (see note at bottom of table)
- b) The next 2 character symbol is a user input, defined at run time, and indicates the vehicle to which the data pertains. Thus, the user can tailor this entry to best describe the vehicle name and/or type. For example, an OA entry could signify Orbiter A.
- c) The last symbol is an two-digit integer indicating the maximum elevation angle for that tracker during the orbit of interest. For example, the ninth row contains a selection for BDQC in orbit 2. BDQC is the acronym for a C-BAND tracker on the island of Bermuda. The entry indicates that the Orbiter vehicle, OA, is in Complex tracking phase (X) and had a maximum elevation angle of 25 degrees.

The symbols are completely defined in a post-script to each table.

ORIGINAL PAGE IS
OF POOR QUALITY

Tracker Support Matrix											
Flight: FLT1						Orbits: 1 - 10					
Cycle: 1						Thu Mar 24 10:14:14 CST 1988					
SITE	1	2	3	4	5	6	7	8	9	10	SITE
TDRE	XOA00	XOA00	XOA00	*OAGG	*OA00	*OA00	*OA00	*OA00	*OA00	*OA00	TDRE
TDRW	XOA00	XOA00	XOA00	*OA00	*OA00	*OA00	*OA00	*OA00	*OA00	*OA00	TDRW
KMRC		XOA25									KMRC
KMTC			XOA25								KMTC
KMAC	XOA25										KMAC
KPTC		XOA25									KPTC
PTPC		XOA25									PTPC
WLPC	XOA25										WLPC
BDQC		XOA25									BDQC
ANTC			XOA25								ANTC
ASCC	XOA25										ASCC
ASTC			XOA25								ASTC
SITE	1	2	3	4	5	6	7	8	9	10	SITE

Vehicles: OA = Orbiter TA = Target OB = Orbiter TB = Target
Selected Sites: * = Non-critical C = Critical X = Complex

AUTOMATION OF THE SPACE STATION CORE MODULE POWER MANAGEMENT
AND DISTRIBUTION SYSTEM

David J. Weeks

National Aeronautics and Space Administration, NASA/MSFC
Marshall Space Flight Center, AL 35812

ABSTRACT

Under the Advanced Development Program for Space Station, Marshall Space Flight Center (MSFC) has been developing advanced automation applications for the Power Management and Distribution (PMAD) system inside the Space Station modules for the past three years. The Space Station Module Power Management and Distribution System (SSM/PMAD) test bed features three artificial intelligence (AI) systems coupled with conventional automation software functioning in an autonomous or "closed-loop" fashion. The AI systems in the test bed include a baseline scheduler/dynamic rescheduler (LES), a load shedding management system (LPLMS), and a fault recovery and management expert system (FRAMES). This test bed will be part of the NASA Systems Autonomy Demonstration for 1990 involving Ames Research Center, Lewis Research Center, Johnson Space Center, and MSFC. This demonstration will feature cooperating expert systems in various Space Station subsystem test beds. Earlier MSFC power system automation efforts contributing to the development of the SSM/PMAD include the Intelligent Data Reduction Expert (I-DARE), the Nickel-Cadmium Battery Expert System (NICBES), and the Autonomously Managed Power System (AMPS) including the fault diagnostic expert system, STARR. It is concluded that advanced automation technology involving AI approaches is sufficiently mature to begin applying the technology to current and planned spacecraft applications including the Space Station.

INTRODUCTION

One purpose of the Space Station advanced development program, implemented FY85-87, was to advance the state-of-the-art of various technologies to enable the development of the planned Station. The specific focus of the SSM/PMAD automation effort was to advance the Space Station Module (laboratory or habitation modules that the crew will live and work in) power system automation capability over previous manned spacecraft power systems [1,2,3].

The short-term objective of the work at MSFC is to develop ground-based knowledge-based systems integrated with actual electrical power system breadboards and test beds to demonstrate the viability of such advanced automation approaches

for spacecraft on-board and ground support applications. It is envisioned that initially such knowledge-based systems would be employed as advisory systems in ground support station roles. When an appropriate confidence level in these systems is reached by program managers, advisory systems would next be employed on-board the spacecraft with the necessary hooks and scars to enable eventual direct control mode implementation. The long-term objective is to develop autonomous operation spacecraft electrical power systems.

In the Electrical Power Branch at MSFC, attention has been focused on comprehensive fault management and dynamic payload rescheduling activities. Comprehensive fault management includes identifying anomalies, diagnosing actual faults (hard and soft), recommending corrective actions for fault recovery, and autonomously implementing fault recovery actions. Dynamic payload rescheduling is necessary in order to operate a closed-loop autonomous electrical power system of significant size and complexity until a new spacecraft baseline schedule is developed and a loads event list is delivered to the power system.

When a fault develops (or begins to develop as an incipient failure) the system should be able to identify what is happening, locate the problem source, recommend actions to be taken, be able to actually implement those actions autonomously, evaluate the payload schedule for perturbations, and reschedule payloads in accordance with the current electrical power system configuration following recovery actions taken by the system. While hard short-circuits in the power system will be handled by the fast and smart switchgear to immediately protect the power system, open-circuits, resistive short-circuits, and impending faults will require intelligent systems to detect and isolate. An intelligent system should be capable of interacting with the power system by opening and closing various switches autonomously in order to narrow the list of malfunctioning component candidates to a minimum number. This implies that the system must know what switchgear it may open and close at any given time. At some point, the system may require a crew member to replace a suspect component to alleviate the problem or further narrow the suspect list. The intelligent system must also be able to discern a sensor failure from a power component failure. If a sensor is deemed to be malfunctioning, the system must

inform the crew that the component should be replaced and that the system will ignore all data from that sensor until it has been informed by the crew that the sensor has been replaced.

The approach within the Information and Electronic Systems Laboratory at MSFC has been to start with well-defined, limited electrical power system applications as stand-alone systems. The next step was to integrate such applications with actual breadboards that are representative of flight systems. The current phase focuses on integrating these knowledge-based systems together and with conventional automation software.

Other research is being conducted in the area of intelligent data reduction for power system telemetry. Such data reduction is important because as much as 95 to 98% of the power system telemetry data at any given moment may be insignificant and manual data reduction is extremely people-intensive and often too late to avoid many problems. Incipient failure detection and other trends analysis for state-of-health monitoring would be greatly facilitated by on-board autonomous data reduction. Future plans involve the development of expert systems for battery management, trends analysis, and on-orbit replaceable unit (ORU) level failure forecasting.

The Electrical Power Branch at MSFC has been involved since 1984 with the development of expert or knowledge-based systems to facilitate the automation of electrical power systems. These knowledge-based systems include the Fault Isolation Expert System (FIES I and FIES II, managed by the Software and Data Management Division in 1982-84), the Space Station Experiment Scheduler (SSES), a fault detection/diagnosis/recovery expert system called STARR, the NICBES for the Hubble Space Telescope power system test bed, the SSM/PMAD system automation product which includes three AI systems [Fault Recovery And Management Expert System (FRAMES), Load Priority List Management System (LPLMS), and Load Enable Scheduler (LES)], the I-DARE, the cooperative expert system project for Scheduling And Fault Analysis/Recovery Integration (SAFARI), and some learning systems research. This paper will focus on the SSM/PMAD automation effort preceded by some discussion of the NICBES and I-DARE projects [4].

EFFORTS BENEFITTING SSM/PMAD

Two MSFC power system automation efforts involving AI are of particular interest to the SSM/PMAD automation effort. These are the NICBES and I-DARE projects.

NICBES

NICBES, the nickel-cadmium battery expert system, is integrated with the Hubble Space Telescope electrical power system test bed, featuring flight-type components. Implemented in PROLOG, the expert system resides on a PC/AT. A block diagram of NICBES and the test bed is given in Figure 1 [5].

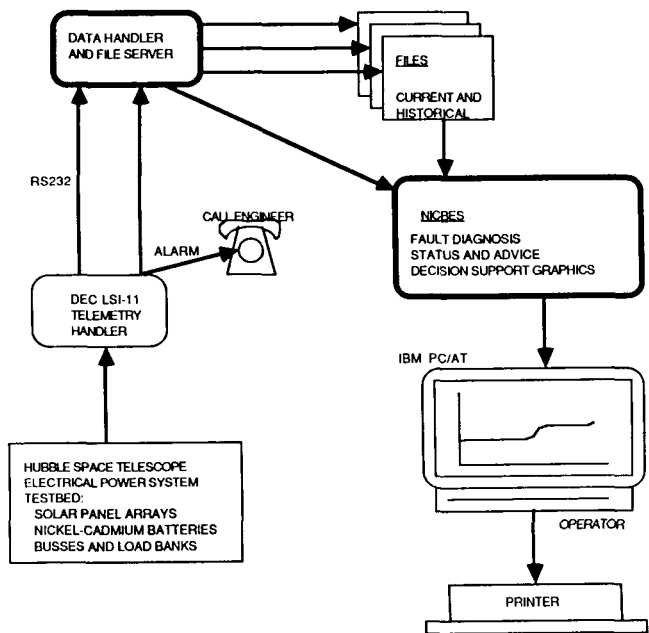


Figure 1. NICBES Functional Block Diagram

This test bed is operated continuously and automatically telephones the test bed personnel and manager at work and at home in the event of an anomaly. When these personnel arrive at the test bed site, they troubleshoot the system and take any steps necessary to restore the system to full operational status while protecting critical flight-type components in the test bed.

NICBES has three major functions in addition to the collection and storage of telemetry data from the test bed. The first function or mode is fault diagnosis. NICBES will independently verify the occurrence of an anomaly, identify the source of the anomaly, and recommend appropriate corrective actions.

The second mode is status and advice. NICBES will evaluate the status of each battery in the test (there are six, 23-cell flight-type batteries) and give advice concerning each battery. Where appropriate, graphs or histograms from the decision support system will be employed to support the advice given. Advice may also be sought on whether the battery is due for reconditioning, a change in workload, or a change in charging scheme.

The third mode is the decision support system which offers twelve plots for any of the six batteries in the system. These plots display summary data that the expert is accustomed to seeing and from which they can verify conclusions drawn by the expert system for the twelve previous simulated orbits.

NICBES is significant in that while only a prototype, it is the only expert system in NASA interfaced with a program-critical electrical power system test bed. The past year and a half of NICBES operation has driven out several features which are to be incorporated in the near future. These include

the upgrading of the number of orbits of data handled from 12 to between 100 and 500; adding an explanation facility; employing a true multi-tasking operating system; and porting to a faster computer (80386 personal computer).

I-DARE

The intelligent data reduction expert system (I-DARE) is implemented on a Lisp workstation in Common Lisp. It is interfaced with the telemetry data stream from the Hubble Space Telescope power system test bed over dedicated lines to the electrical power systems test beds laboratory in another building and will autonomously reduce the data to the significant components. The concept of such data reduction is shown in Figure 2. Such data reduction is currently extremely slow and manhour intensive. This is a two year research effort with emphasis on emulating in an expert system the behavior of the test bed engineers in reducing the power system telemetry data to its significant components [8].

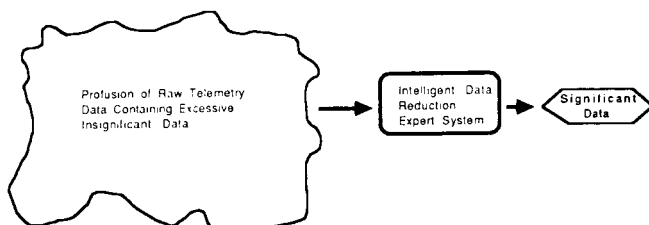


Figure 2. Intelligent Data Reduction Expert System

SSM/PMAD AUTOMATION

The Space Station advanced development breadboard for power system automation at MSFC is illustrated in Figure 3. The SSM/PMAD is a dual channel 20 kHz power system sized large enough to operate a substantial number of realistically sized loads simultaneously and autonomously. The architecture and functionality are based upon the requirements of a Space Station Core Module. Autonomy is pushed down to the lowest levels, the lowest level processors (LLP) located in the load centers and subsystem distributors. The Communications and Algorithmic Controller (CAC) implemented on a VME/10 mini-computer performs numeric computations, controls the breadboard communications (Ethernet and RS-422 serial data networks) and directs the LLPs. The breadboard includes three AI systems which function in a cooperative mode. These are the LPLMS, the LES, and the FRAMES [6].

LPLMS

The prioritization expert system is called the loads priority list management system (LPLMS) and keeps up with the dynamic priorities of all payloads while developing current global load shedding lists for the SSM/PMAD every 15 min in preparation for contingencies which necessitate load shedding. The global load shedding list is broken down by the CAC into local load shedding list for each load center/subsystem distributor. The LPLMS is implemented in Common LISP and resides on a LISP workstation.

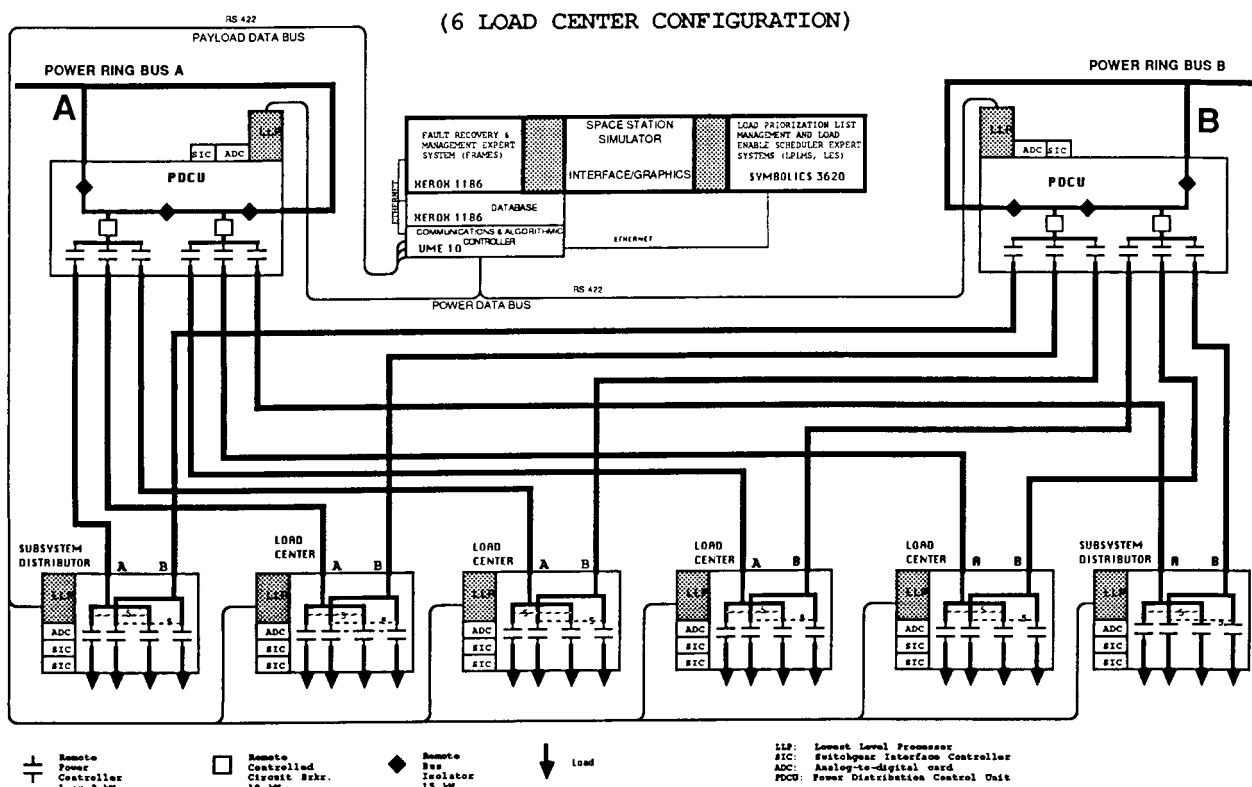


Figure 3. SSM/PMAD Block Diagram

ORIGINAL PAGE IS OF POOR QUALITY

LES

The scheduler expert system is called the load enable scheduler (LES) and schedules/reschedules the payloads for the Space Station module. LES is a special version of MAESTRO, the Martin Marietta IRAD AI scheduling effort. The baseline scheduling capability of LES generates a baseline schedule in the form of event lists which provide the SSM/PMAD test bed with operational scenarios. LES can handle hundreds of scheduling constraints. It can reschedule the loads in response to a reconfiguration of the SSM/PMAD (due to a fault, change in power allocation, or operator instructions) in as little as five minutes. This allows the SSM/PMAD to continue servicing loads in accordance with mission objectives and priorities the best that it can under deteriorating circumstances. This resource scheduler also resides on the Lisp workstation hosting the LPLMS and is coded in Lisp [7].

FRAMES

The fault recovery and management expert system (FRAMES) resides on a separate Lisp workstation and is implemented in the Common List Object System (CLOS). This expert system watches over the entire breadboard operation looking for anomalies and impending failures. The functionality of FRAMES actually extends to the lowest level processors and the smart switchgear with their associated controllers in the breadboard for comprehensive fault management of the entire breadboard.

FRAMES is responsible for detecting faults, advising the operator of appropriate corrective actions, and in many cases autonomously implementing corrective actions through power system reconfiguration. The expert system will carry out trends analysis seeking incipient failures and soft shorts as well as open circuits. Fast response remote power controllers (RPC) respond to the hard system shorts.

When a fault or anomaly occurs in this Space Station module power system breadboard, FRAMES detects, diagnoses, and recommends corrective actions (in the case of critical loads, it also autonomously performs corrective actions). Then the LES interface determines if a new payload schedule is necessary and if so, directs LES to reschedule the payloads. The LPLMS derives new load shedding lists from the new schedule and issues these lists to the CAC which develops local load shedding lists for LLPs in the load centers and subsystem distributors, thus completing the closed automation loop as illustrated in Figure 4.

SPECIFIC RESULTS

NICBES is employed daily and has proved to be a valuable aid to the Hubble Space Telescope power system breadboard personnel. Its major deficiency resides in the single tasking mode of the operating system on the PC/AT. An upgrade in progress will move NICBES to a 80386-based computer system. The revised NICBES will involve a multi-tasking operating system which will allow continuous data handling. It will increase the number of orbits data handling from 12 to

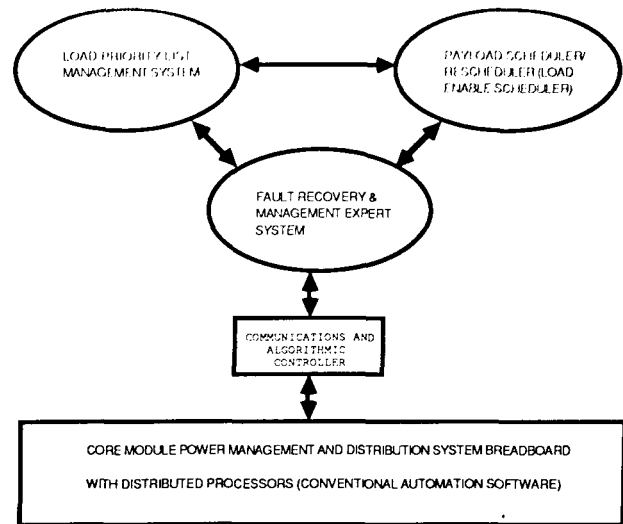


Figure 4. Closed-Loop Operation

between 100 and 500 while adding a natural language explanation facility. An editor to facilitate the rule base maintenance is also under development. Plans are now being formulated to develop another expert system for a new nickel-hydrogen battery test bed for the Hubble Space Telescope program.

I-DARE is a simple model at this point and will continue to be under development for the next year. The data link for the AI workstation is in place and intensive interviews with the domain experts (the data reducing engineers) have taken place. A prototype has been developed which does some telemetry transmission error detecting and reduces the actual telemetry data to a limited extent. The goal is to at least match human performance in this phase.

Prototypes of LPLMS, LES, and FRAMES have been completed. As this paper is being written, these systems are being integrated with one another and with the SSM/PMAD test bed hardware and conventional software. A full-up integrated test will be conducted at MSFC in August 1988. Work has already commenced on further development of the SSM/PMAD to support the 1990 NASA Systems Autonomy Demonstration featuring cooperating expert systems in Space Station subsystem test beds.

CONCLUSIONS

Artificial Intelligence systems will be employed in the automation of future spacecraft electrical power systems including the Space Station. Initially, these systems will serve in advisory roles primarily in the ground support stations with some limited advisory roles on-board manned spacecraft. The hooks and scars will be designed in place for these advisory systems to evolve to direct control implementation modes of operation. Initially, such control systems may be placed primarily in the ground support stations until program management, ground support personnel, and crew confidence is established to the point that they may be implemented on-board. On-board AI control systems will be implemented such

that if the AI system fails, it will not cause a catastrophic situation but will instead allow the subsystem to continue operations in a less automated but safe fashion.

The eventual goal is completely autonomously operated spacecraft electrical power systems. Conventional automation software approaches have too many gaps to provide overall autonomy. Utilizing AI approaches allows for comprehensive fault management and dynamic rescheduling capabilities for the electrical power system. Autonomous intelligent data reduction will enable enhanced state-of-health monitoring and better trends analysis including incipient failure detection which will enable recovery actions to be taken in order to preclude an actual failure. Many of the approaches and techniques developed to support autonomous electrical power systems operation may also be utilized for other spacecraft subsystems as well.

ACKNOWLEDGMENTS

The author wishes to acknowledge the technical support of the knowledge-based projects discussed in this paper by the Martin Marietta Corporation, the University of Alabama in Huntsville, and the personnel of the Electrical Power Branch at MSFC. It is also acknowledged that the reverential awe of the Lord God is the beginning of all knowledge (Proverbs 7:1).

REFERENCES

1. Weeks, D. J., and Bechtel, R. T.: "Autonomously Managed High Power Systems," Proceedings of the 20th IECEC, Miami, FL, 1985.
2. Weeks, D. J.: "Application of Expert Systems in the Common Module Electrical Power System," SPIE Vol. 580 Space Station Automation, Cambridge, MA, 1985.
3. Weeks, D. J.: "Expert Systems in Space," IEEE Potentials, Vol. 6, No. 2, 1987.
4. Weeks, D. J.: "Artificial Intelligence Approaches in Space Power Systems Automation at Marshall Space Flight Center," Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, TN, 1988.
5. Kirkwood, N., and Weeks, D. J.: "Diagnosing Battery Behavior with an Expert System in PROLOG," Proceedings of the 21st IECEC, San Diego, CA, 1986.
6. Weeks, D. J.: "Artificial Intelligence and Space Power Systems Automation," Proceedings of the Third Conference on Artificial Intelligence for Space Applications, Huntsville, AL, 1987.
7. Geoffroy, A. L., et al.: "Power Resource Management Scheduling for Scientific Space Platform Applications," Proceedings of the 22nd IECEC, Philadelphia, PA, 1987.
8. Weeks, D. J.: "Space Power System Automation Approaches at the George C. Marshall Space Flight Center," Proceedings of the 22nd IECEC, Philadelphia, PA, 1987.

EMMA : THE EXPERT SYSTEM FOR
MUNITION MAINTENANCE

Barry E. Mullins, Capt, USAF
Air Force Armament Laboratory (AFATL/FXG)
Eglin Air Force Base, Florida 32542-5434

ABSTRACT

EMMA (Expert Missile Maintenance Aid) is the result of research sponsored by the Air Force Armament Laboratory, Air Force Systems Command at Eglin Air Force Base (AFB), Florida. It is a first attempt to enhance maintenance of a tactical munition at the field and depot level by using artificial intelligence (AI) techniques. The ultimate goal of EMMA is to help a novice maintenance technician isolate and diagnose electronic, electromechanical, and mechanical equipment faults to the board/chassis level more quickly and consistently than the best human expert using the best currently available automatic test equipment (ATE). To this end, EMMA augments existing ATE with an expert system that captures the knowledge of design and maintenance experts.

This paper describes the EMMA program. It addresses such issues as how the field-level expert system prototypes were evaluated as well as the results of the evaluations. Additionally, current work on the depot-level prototypes is discussed as well as issues related to using DOD-STD-2167 to document the development of expert systems. This paper will briefly address several study tasks performed during EMMA. The paper concludes with a discussion of future plans for a follow-on program and other areas of concern.

INTRODUCTION

Weapon systems of today are undoubtedly benefiting from technology advances and justifiably so. Munitions are becoming more sophisticated and "smarter" as a result of this technology. Electronically sophisticated munitions are quickly infiltrating the Department of Defense arsenal of weapons. Simple bombs are becoming relics of the past.

However, some shortcomings can be associated with incorporating new technology into current and future weapon sys-

tems. Munition maintenance will surely suffer as a consequence of this technology. The technology advances that have improved the effectiveness of munitions are simultaneously complicating the maintenance of these munitions by increasing the functionality of the munition typically making the munition harder to maintain. Munition test equipment and associated software do not adequately diagnose faults. Automatic test equipment (ATE) is plagued by high false alarm rates. Guidance and control sections returned to the depot are currently experiencing approximately 28 to 44% retest OKs. Many faults, as many as one out of every four, cannot be detected by ATE. Sequential testing is typically how ATE performs diagnostic testing. This limits the diagnostic capabilities of the test equipment. Additionally, ATE cannot diagnose beyond multiple linked components.

Another aspect of munition maintenance that must be addressed when dealing with munition reliability is the personnel shortage. The current shortage of skilled munition maintenance technicians is a serious problem. Demographic projections indicate that this dilemma will not subside in the near future. Since experienced technicians are able to diagnose a fault quicker and more reliably than a novice, the knowledge acquire by the experienced technician (expert) throughout the years should be captured so that this knowledge can be used by novice technicians during future diagnostic sessions.

Artificial intelligence (AI) technology is one approach to increasing the reliability and maintainability of existing and future weapon systems. One popular and heavily cited definition of artificial intelligence is provided by Dr. Elaine Rich, University of Texas at Austin. She defines AI as follows: "Artificial intelligence is the study of how computers do things at which, at the moment, people are better" (Rich, 1983:1). A subset of AI is a field called expert

systems. This area of AI has emerged recently with the greatest amount of success (Hayes-Roth et al., 1983:xi). Donald Waterman defines expert systems as "sophisticated computer programs that manipulate knowledge to solve problems efficiently and effectively in a narrow problem area" (Waterman, 1986:xvii).

The Air Force has recognized the importance of increasing munition reliability. In a joint memorandum, General Gabriel, former Air Force Chief of Staff, and Verne Orr, former Secretary of the Air Force stated, "For too long, the reliability and maintainability of our weapon systems have been secondary considerations in the acquisition process. It is time to change this practice"

EMMA

EMMA (Expert Missile Maintenance Aid) is the result of research sponsored by the Air Force Armament Laboratory, Air Force Systems Command at Eglin AFB, Florida and is a first attempt to enhance Air Force tactical munition maintenance by applying artificial intelligence/expert system technology to ATE. The objective of EMMA is to develop an automated smart munition test system that augments existing ATE. The ultimate goal of EMMA is to help a novice munition technician isolate and diagnose electronic, electromechanical, and mechanical equipment faults to the board/chassis level more quickly and consistently than the best human expert using the best currently available ATE.

EMMA is a thirty month effort split into two phases. Phase 1 began in September 1986 and concluded 10 months later in July 1987. This phase addressed the field-level maintenance of tactical munitions and ultimately resulted in two field-level expert system prototypes. Phase 2 began in August 1987 and is scheduled to conclude in April 1989, 20 months later. Phase 2 focuses on depot-level maintenance and will produce two depot-level expert system prototypes. Since depot-level diagnostic activities are more in-depth and detailed than the field, this phase is expected to be more difficult and of greater complexity. This accounts for the greater time allotted to this phase. The prototypes from both phases are targeted towards the maintenance technicians. Since EMMA is constrained by schedule and money, the number of tests developed under this effort is limited, yet sufficient to demonstrate concept feasibility of using expert systems for munition maintenance.

EMMA draws on many different types of knowledge and information to perform the diagnosis of the faulty munition. The EMMA knowledge base consists of maintenance rules or Technical Orders (TOs),

maintenance technician practices (heuristics), Unit Under Test (UUT) design, existing test equipment capabilities, failure rates, and test costs. This knowledge is gleaned from various sources including TOs, schematics of the UUT and the test equipment, knowledge acquisition interviews with munition maintenance experts and the experts that designed the munition and test equipment. Figure 1 depicts how this knowledge is brought to bear on the problem of diagnosing the faulty munition. First, the symptoms are derived from the test equipment and technician observations. This information is supplied to the expert system via a sophisticated, user-friendly interface. The expert system then employs the knowledge stored in the knowledge bases and derives a repair strategy which is displayed to the technician using the EMMA computer.

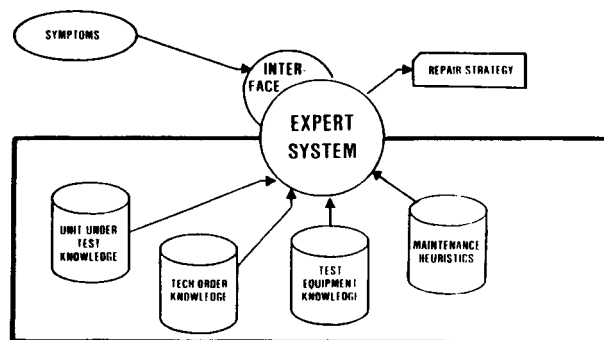


Figure 1. EMMA Expert System

EMMA is a dual contract effort performed by Raytheon Company, Missile Systems Division in Bedford Massachusetts, and Rockwell International Corporation, Autonetics Sensors and Aircraft Systems Division in Anaheim, California. Both contractors will develop a phase 1 and phase 2 EMMA prototype resulting in a total of four prototypes. Both contractors were allowed to select their candidate vehicle for the EMMA program within specified limits. Raytheon selected the AIM-7F Sparrow missile as their candidate munition. Rockwell chose the GBU-15 modular glide bomb.

THE AIM-7F FIELD-LEVEL EMMA PROTOTYPE

The Raytheon field-level (phase 1) EMMA prototype was designed to enhance the field-level maintenance of the AIM-7F missile by augmenting the missile's test set. All references to the word "EMMA" in this section refer to the Raytheon AIM-7F version of EMMA. The field-level test set for the AIM-7F is the AN/DSM-162 test set. EMMA is hosted on a Symbolics 3670 LISP machine running the expert system shell ART (Automated Reasoning Tool). ART pro-

vides a production language that is primarily rule-based. Consequently, EMMA was developed using the rule-based approach. The Symbolics computer is connected to the AN/DSM-162 test set via an RS-232 cable. Figure 2 illustrates the major components of the EMMA system and how they are interconnected.

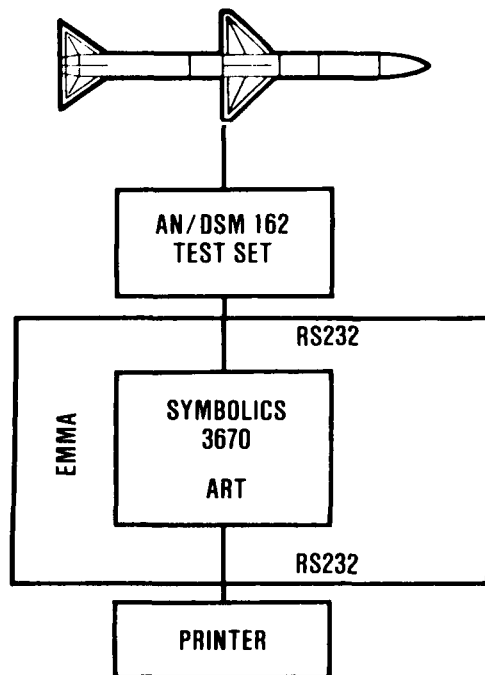


Figure 2. AIM-7F Field-level EMMA Prototype

The RS-232 cable allows EMMA to operate in three modes -- automatic, semi-automatic, and manual. The distinguishing characteristics of these modes is the level of automation EMMA is allowed during the diagnostic session. The automatic mode uses the RS-232 interface to allow EMMA to direct the diagnostic testing and resequencing of tests. EMMA automatically accepts data from the test set via the RS-232 cable, performs the fault isolation, and directs the test set to perform additional test, if required, until the fault is detected or all tests pass. If a fault is detected during automatic operation, the user may switch to semi-automatic mode for closer control over the testing and the ability to query after each test segment.

The semi-automatic mode operates similar to the automatic mode with one exception. This mode stops execution of EMMA at the completion of each unique test segment. This allows the technician to query EMMA recommendations using the explanation capability. Another advantage of the automatic modes (semi and full) is

data integrity. Since EMMA passes the data between the test set and the Symbolics computer via the RS-232 cable, the data are more likely to remain valid as opposed to transferring data via a technician who could inadvertently introduce errors.

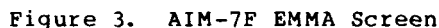
The last mode is manual. This mode is provided in case an RS-232 connection is not possible. As the name implies, all activities between the test set and the Symbolics computer must be performed manually by the technician. EMMA will direct the technician to perform the appropriate actions to the test set and wait for the response. The technician enters the responses from the test set into EMMA.

As with most expert systems, EMMA is able to explain its reasoning process to the user (technician). EMMA explains its fault detection and resequencing logic. In other words, EMMA explains a detected fault and why a certain test is being recommended. Two levels of explanation are available depending upon the experience of the technician. The technician may request an explanation during any phase of the diagnostic process. This allows the technician to query EMMA during a consultation which heightens the technician's understanding of what EMMA is doing while simultaneously providing the technician with a valuable training aid.

One of the most critical aspects of any software system is its user-friendliness. If the system is difficult to use and the user does not use it, it has failed. EMMA uses windows to relay information to the technician and accepts information via menus. Using a mouse, the technician is able to enter data quickly and accurately without having to learn cryptic commands. The majority of the data entered into EMMA by the technician is done using the mouse; however, some keyboard input is required. Figure 3 shows the screen of a Symbolics computer running EMMA.

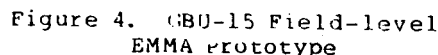
THE GBU-15 FIELD-LEVEL EMMA PROTOTYPE

The Rockwell field-level (phase 1) EMMA prototype was designed to enhance the field-level maintenance of the GBU-15 glide bomb by augmenting the field-level test set -- GJM-55. All references to the word "EMMA" in this section refer to the Rockwell GBU-15 version of EMMA unless stated otherwise. EMMA is hosted on a IBM PC/AT compatible computer running the expert system shell M.I. Although, the M.I language is primarily rule-based, EMMA was developed using an object oriented approach. The rules of the knowledge base reference objects and object attributes. This EMMA did not support the capability for an automatic mode due to hardware



Uncertainty is addressed in this version of EMMA. When EMMA asks the technician for information, the technician may enter "unknown" as a response. EMMA will accommodate this response by adapting its reasoning process using uncertainty. Uncertainty is handled using a MYCIN-like representation. When a recommendation is displayed to the technician, the certainty of the recommendation is also displayed to indicate the belief of the recommendation.

EMMA exploits the use of pull-down menus and function keys on the computer to



make it as user friendly as possible. The majority of technician interaction with EMMA is performed using the keyboard. The technician typically responds to EMMA questions and requests with short answers thereby reducing the probability of erroneous data being entered. Figure 5 shows the screen of the computer running EMMA.

EVALUATION OF THE EMMA PROTOTYPES

Meaningful evaluation of expert systems has been an often discussed but seldom achieved topic within recent years. More often than not quantitative metrics are simply not available or meaningful as an evaluation measure. Since an expert system encapsulates the knowledge of a given expert in a given field, the effective evaluation of the expert system may be difficult at best. Validation must be used to justify the representation levels of expert systems (O'Keefe et al., 1987).

Validation is typically considered a part of evaluation, and evaluation is concerned with determining the comprehensive value of an expert system (O'Keefe et al., 1987). Validation should not be confused with verification. "Validation refers to building the right system (that is, substantiating that a system performs with an acceptable level of accuracy), whereas verification refers to building the system 'right' (that is, substantiating that a system correctly implements its specifications)" (O'Keefe et al., 1987). Validation and verification will be addressed in this paper as they apply to EMMA.

Verification of EMMA

A unique aspect of the verification of the EMMA program is that it uses DOD-STD-2167, the Defense System Software Development standard, to develop the expert system prototypes. This is one of the first attempts to apply this standard to the development of an expert system. DOD-STD-2167 traditionally employs top-down development of large software systems. Expert systems on the other hand are developed using a development methodology that is less rigidly defined which typically entails an iterative approach to software development. Additionally, expert systems are usually created by a relatively small team.

The EMMA program has shown that expert systems can be developed using DOD-STD-2167 software development requirements. With some careful tailoring of some of the documents, this standard can be effectively used to document the program and provide the program manager valuable insight into the contractor's software development, testing, and evaluation efforts. The tailored documents were altered to accommodate the iterative nature of expert system development.

Since verification must determine whether an expert system correctly implements its specifications, testing must occur in order to validate this requirement. Again, DOD-STD-2167 proved to be adequate for verification testing once extended. Using the testing procedures called out in this standard, the correct

EMMA CONFIG NO: 33 TEST NO: 0096 NOGO: INV/M6 A1

<u>SRU</u>	<u>COMPONENT</u>	<u>CABLE</u>	<u>CERTAINTY</u>
INV/CONV			60%
CONTROL UNIT (HARNESS)	UUT INPUT CABLE	053A20(BLU)	40%

RECOMMENDATION:

CHECK THE INPUT CABLE OF THE UNIT UNDER TEST. IF OK, THEN
R&R INV/CONV

JUSTIFICATION:

A RELATED TEST WITH A DIFFERENT INPUT CHANNEL PASSED AND THE PARAMETERS THAT FAILED WERE APPROXIMATELY ZERO; THUS THE INPUT CABLES OF THE UNIT UNDER TEST COULD BE THE CAUSE. A MORE LIKELY FAILURE IS THE INV/CONV BECAUSE THE +28 VDC ELECTRONICS IS COMPLEX AND COULD CAUSE ZERO FAILURES AS INDICATED

Figure 5. GBU-15 EMMA Screen

implementations of the specifications for EMMA were verified. Two levels of testing occurred to accomplish this task. First, informal testing took place. This testing verified the integrity of the individual computer software units before the units were integrated into the system and tested as a system. Informal testing was performed by the knowledge engineer. Since expert system development is iterative in nature, informal testing essentially occurs throughout development. The knowledge engineer and the expert verify the expert system and identify potential corrections and enhancements. Based on these recommendations, the knowledge engineer was able to incorporate the recommendations. Second, formal testing occurred. An independent team performed the formal testing by exercising EMMA using test plans and test descriptions generated using DOD-STD-2167.

Validation of EMMA

The validation of EMMA will be addressed in this paper in two areas. First, the performance validation of EMMA will be discussed (i.e., how well EMMA performed). Second, the human factors aspect of validation will be addressed. Both areas are extremely important to the success of an expert system. The following paragraphs will present the validation of the two EMMA prototypes. The validation methodology will be discussed followed by the results of the validation.

As with most expert systems, the ultimate measure of success is determined when the system is used by the end users (in this case the field-level munition technician). This is the approach taken with the EMMA program. Both contractors took their respective prototypes to Air Force bases in which their selected munition is used and a field-level maintenance capability exists. This allowed the prototypes to be evaluated in an actual field test environment.

After considering several alternatives, both contractors decided to use a toggle switch box to insert faults into a known good missile. This approach was necessary since it was feared that the maintenance squadrons in the field might not have a sufficient number of faulty munitions during the evaluation period. These faults were induced by the user by simply toggling one of the switches which in turn would disturb one or more signals within the munition. The faults were defined by the domain expert such that the faults would adequately exercise the various characteristics of the EMMA prototypes which included the resequencing logic, the explanation capability, and the fault isolation logic. To verify the expert was not trying to select only the

faults EMMA could handle best, the maintenance experts agreed the selected faults were representative of faults they commonly experience.

The AIM-7F EMMA Evaluation. Raytheon took their field level AIM-7F prototype to the 325th Equipment Maintenance Squadron (EMS) at Tyndall AFB, Florida for an evaluation period that began on 8 June 1987 and concluded on 12 June 1987.

The argument could be made that EMMA should accurately diagnose all the induced faults since the expert system and the faults were derived from the same source -- the domain expert. In order to demonstrate the robustness of EMMA, an additional evaluation methodology was used. Two faulted missiles were saved by the EMS prior to the evaluation. These missiles had previously failed testing using the AN/DSM-162 test set. However, the fault data for these missiles were not released by the EMS personnel until after the EMMA evaluation. A third missile became available during the EMMA evaluation by failing a flight line test during prelaunch tuning. This missile was an excellent exercise for the EMMA prototype since it was not previously tested by the AN/DSM-162 test set. Its fault was unknown to everyone present at the evaluation. All three missiles (sometimes referred to as "mystery missiles" due to their unknown past) contained faults unknown to EMMA or the domain expert thereby exercising EMMA in an unpredictable manner.

Four munition maintenance technicians from the EMS at Tyndall were used for the evaluation. Two technicians were classified as novice with little experience with the AN/DSM-162 test set and its associated operating procedures. The other two technicians were classified as experts with a substantial background in using the AN/DSM-162 test set. Two teams of two technicians were created consisting of one expert and one novice. One team (hereafter referred to as the EMMA team) received extensive training on the operation of EMMA. The other team (hereafter referred to as the non-EMMA team) was not trained on the EMMA system and served as a baseline for the evaluation.

Twelve faults were inserted into a known good missile using the toggle switch box. The faults were diagnosed by the EMMA team using the EMMA system and the non-EMMA team using just the AN/DSM-162 test set and the applicable TO. Performance of the two teams was based on the level of expertise of the operator, duration of test, and the ability to diagnose the fault accurately.

ORIGINAL PAGE IS OF POOR QUALITY

The results of this evaluation exercise were very promising. There were three significant results derived from the evaluation. First, the EMMA system operated by the EMMA team was able to consistently diagnose the fault quicker than the non-EMMA team using just the AN/DSM-162 and the TO regardless of the experience level of the EMMA operator. A time savings of 20% was seen with the novice using EMMA over the expert using the AN/DSM-162. Second, novice technicians using the EMMA system significantly outperformed (better fault diagnoses) novice technicians using just the AN/DSM-162 and performed 33% better than expert technicians using just the AN/DSM-162. Finally, EMMA's explanation capabilities significantly enhanced the abilities of the EMMA team to determine the reason behind each fault.

Once EMMA's abilities were exercised using the induced faults, EMMA was pitted against the mystery missiles again with excellent results. The EMMA team using EMMA correctly isolated the faults in all three missiles. Only after EMMA diagnosed the faults was the previous testing data on the missiles released. EMMA's diagnosis was consistent with this data.

User acceptance of EMMA was outstanding. In fact, the technicians accepted EMMA's diagnosis of the missile from the flight line and said they would have, if allowed, sent the missile to the depot with no further testing using the AN/DSM-162 test set. This exemplifies EMMA's acceptance by the EMS maintenance personnel at Tyndall AFB. The technicians found the system to be very user-friendly. The mouse and the use of menus made the system easier to use than the bulky and cumbersome TOs. Also, the explanation capability proved to be an effective training mechanism.

The GBU-15 EMMA Evaluation. Rockwell evaluated their GBU-15 EMMA at the 4th Equipment Maintenance Squadron located at Seymour Johnson AFB, North Carolina during the period of 22 June through 29 June 1987.

Four maintenance technicians were used in the evaluation of the EMMA prototype. Two technicians were considered experts with several years of experience with the GBU-15 test environment. The remaining two technicians were considered novices with less than 6 months of experience. Another important distinction between the expert and novice technicians is the fact that the expert technicians owned personal computers and therefore were familiar with how computers operate whereas the novice technicians did not own computers and had never used a computer before the EMMA evaluation. All four technicians were trained on how to use the

EMMA system. After this brief training, the technicians felt very comfortable using the system.

Twenty-two simulated faults were induced into the known good munition with the intent of evaluating EMMA's capabilities to handle the following five areas: resolution of ambiguities between major shop replaceable units (SRU), referencing lower configuration testing to facilitate further component resolution, distinguishing between a cable failure and a circuit card assembly (CCA) gain failure, resolution of ambiguities between CCA's, and recognizing operator errors or test set problems. Six of the twenty-two induced faults were in the all-up-round (AUR) configuration (i.e., the test was performed while the GBU-15 munition was completely intact). The remaining sixteen faults were in the control module stand alone configuration. EMMA was able to handle these five areas by analyzing additional test parameters as well as instituting and analyzing tests related to the failed test.

The diagnostic results of the induced faults showed substantial time savings in fault isolation and increased diagnostic capabilities. While the munition was in the control module stand alone configuration, a time savings of 40% was seen over conventional testing with the GJM-55. When the munition was in the AUR configuration, EMMA was able to provide up to 74% time savings. This is due to EMMA's capability to resolve failures while the munition is in the AUR configuration thereby saving the technician from having to perform testing in stand alone configuration.

The GJM-55 test set, in some situations, will recommend more than one suspected failure. This group of failures is called an ambiguity group since the test set cannot resolve any further than this group. This is another of EMMA's capabilities that demonstrated promising performance as seen by the results that follow. EMMA also considered the possibility of a cable harness failure or the test set is failing. Based on these capabilities EMMA was able to significantly improve fault isolation. The results of the twenty-two simulated faults demonstrate this improvement. EMMA added a wiring harness check to 50% of all tests. EMMA deleted a CCA from an ambiguity group 40% of the time thereby reducing the number of CCA to be considered during testing. EMMA added a CCA to an ambiguity group 30% of the time to insure all potential CCA's are considered during the testing. This indicates that the test set sometimes did not consider all potential CCA's. Finally, EMMA exchanged one suspect CCA in an ambiguity group for another CCA 10% of the

time. The ability to manipulate the ambiguity group to benefit fault isolation was demonstrated by EMMA and proved to be an effective fault isolation technique. These results directly support the time savings previously mentioned.

The GBU-15 EMMA prototype also received accolades for its user friendliness. The technicians used EMMA with comfort and found several items to be particularly laudable. Among these items was the understandability of EMMA. The explanation capability provided easy to understand responses. Another aspect they found beneficial was the addition of the internal wiring harness check as one of the reasons for a fault since this check is relatively "inexpensive" to perform and can prevent unnecessary and potentially costly future testing. The training potential of EMMA was also mentioned as one of its major assets with the shortage of skilled technicians in the munition maintenance field.

EMMA PHASE 2

Both contractors are currently in phase 2 of the EMMA program. As previously mentioned, phase 2 focuses on the maintenance of tactical munitions at the depot level. More specifically, Raytheon is focusing on the depot-level maintenance of the AIM-7F. Rockwell is using the GBU-15 as its depot-level maintenance munition. Phase 2 is a natural extension of phase 1 since field-level faults are sent to the depot for repair. The prototypes developed during the phase 2 effort will be more detailed extensions of the phase 1 prototypes with one exception; the phase 2 prototypes will augment the depot-level test sets. The depot test set for the AIM-7F is the AN/DPM-22 test set. The GBU-15 depot-level test set is CATS (Calculator Automatic Test Station).

The depot-level prototypes will be implemented on the same computer hardware using the same expert system shells as the field-level prototypes. However, one difference between the field and depot prototypes for both contractors is the interface between the test set and the EMMA computer. The Raytheon interface will only support one-way communication from the test set to the EMMA computer due to test set limitations. This is different than the two-way communication of the field prototype. Rockwell is using a two-way communication interface between the test set and the EMMA computer whereas the field prototype interface was manual. Both prototypes will again incorporate an explanation capability for the technicians.

The evaluation of the depot prototypes will follow the same methodology used in phase 1. Each prototype will be evaluated at the actual depot location by actual depot technicians. Once again, both evaluations are scheduled to last five days and are scheduled to occur in February 1989. The AIM-7F prototype will be evaluated at the Naval Aviation Depot (NAVAVNDEP) in Alameda, California. The GBU-15 prototype will be evaluated at Rockwell's Missile Systems Division in Atlanta, Georgia since an organic depot capability currently does not exist.

FOLLOW-ON PROGRAMS TO EMMA -- EMMA 2

A follow-on program will be initiated in early 1990 -- EMMA 2. The primary thrust of EMMA 2 is to develop an expert system that is capable of diagnosing a family of tactical munitions at the depot level. The current EMMA is limited to one munition per prototype. EMMA 2 would attempt to expand the current prototype capabilities to include multiple munitions from the same family (e.g., AIM family, GBU family, surface-to-air family, etc.). EMMA 2 would draw on the best features of all prototypes developed in the two phases of EMMA to derive a robust system.

OTHER ISSUES/OBSERVATIONS

The following paragraphs present other areas of interest to the EMMA program.

Studies

As part of the phase 1 effort, five studies were conducted to address areas of concern that could be incorporated into the depot-level prototypes and potentially in future maintenance expert systems. These studies included the reuse of munition test programs, the use of the Ada language for expert system development and ATE test programs, the applicability of the Modular Automatic Test Equipment (MATE) standard to EMMA, the applicability of the Warner Robins Reliability Asset Monitor (RAM) database to EMMA and future maintenance expert systems, and the security issues of expert systems. A complete discussion of the results of these studies is beyond the scope of this paper. Therefore, the interested reader is referred to the two final reports of phase 1 (Elerin et al., 1987; Davis, 1987). These final reports are split into two volumes; the second volume contains a complete discussion of the results of these studies.

Current Maintenance Philosophy

The current munition maintenance philosophy of the Tactical Air Command (TAC) for field maintenance is that of fault detection (go/nogo testing). If a fault does occur in the field, the suspected

ORIGINAL PAGE IS OF POOR QUALITY

faulty section of the missile is sent to the depot for repair. One of the driving factors of this philosophy is the shortage of skilled maintenance technicians in field-level maintenance.

Since training these technicians is costly, TAC decided to eliminate an Air Force Specialty Code (AFSC) for munition maintenance. The deleted AFSC, 316X1L, was an electronics munition maintenance specialist. With this specialist no longer available, munition, not electronic munition, specialist are diagnosing today's munitions. This tends to create problems. The munition specialists are typically not adequately trained to diagnose the electronically-sophisticated munitions of today.

EMMA is capable of providing the necessary training of munition maintenance technicians. Using the explanation capabilities of EMMA, a technician can quickly become skilled at diagnosing the munition. Since EMMA's knowledge is gleaned from diagnostic and design experts, the novice munition technician using EMMA will effectively be performing as if he has an expert maintenance technician, the designer of the test set, the munition designer, and an instructor looking over his shoulder during the diagnosis. Another aspect of EMMA that should reduce overall maintenance costs is its ability to diagnose to a greater component level than existing test sets used by today's technicians. This should significantly reduce the costs associated with shipping faulty munitions to the depot, since more faults can be isolated at the field.

The munition depots are currently responsible for the majority of munition repairs. The field has very limited repair capabilities. This results in increased costs for overall maintenance of a munition system. The obvious cost is transportation of the munition between the field and the depot. However, the less tangible and potentially more significant cost is that of having the munition out of the inventory. This effectively reduces the number of missiles available for exercises or conflict.

Future Maintenance Systems

EMMA prototypes have proved the feasibility of applying AI to munition maintenance. In future weapon systems, maintenance expert systems should evolve with the weapon system instead of after the fact. This would allow the expert system to capture knowledge about the weapon as it is developed. Furthermore, the expert system should be incorporated directly into the ATE instead of augmenting the ATE with a separate computer system. This should make weapon systems of the future

more supportable by considering the maintenance aspect early in the weapon life cycle.

SUMMARY

Tactical munition maintenance of today has problems. EMMA is an attempt to relieve some of these problems by applying artificial intelligence/expert system technology. The results of EMMA indicate that this approach to munition maintenance has significant potential for future tactical maintenance systems.

Corporate knowledge retention is one of the premium benefits of EMMA. Since EMMA is updated easily and it never "forgets" knowledge, EMMA is an excellent tool for storing corporate knowledge as technicians come and go. Also, EMMA provides consistent, high quality diagnosis since it never has a "bad" day as contrasted with technicians. Rapid fault isolation and efficient manpower utilization are two more benefits of using EMMA. These benefits provided by EMMA will result in substantial mission payoffs. Weapon system downtime will be decreased as well as personnel requirements and training time. However, the most significant payoff is the increase in the reliability of munition maintenance procedures.

BIBLIOGRAPHY

- Davis, Kathy. Expert Missile Maintenance Aid (EMMA) - Rockwell, Final Report for Period September 1986 - July 1987. Air Force Armament Laboratory Technical Report AFATL-TR-87-43, Eglin AFB, Florida, October 1987.
- Elerin, Liese M. Expert Missile Maintenance Aid (EMMA), Final Report for Period September 1986 - July 1987. Air Force Armament Laboratory Technical Report AFATL-TR-87-46, Eglin AFB, Florida, November 1987.
- Hayes-Roth, Fredrick et al. Building Expert Systems. Reading Massachusetts: Addison-Wesley Publishing Company, 1983.
- O'Keefe, Robert M. et al. "Validating Expert System Performance," IEEE Expert, 2: 81-89 (Winter 1987).
- Rich, Elaine. Artificial Intelligence. New York: McGraw-Hill Book Company, 1983.
- Waterman, Donald A. A Guide to Expert Systems. Reading Massachusetts: Addison-Wesley Publishing Company, 1986.

DIAGNOSTICS IN THE
EXTENDABLE INTEGRATED SUPPORT ENVIRONMENT (EISE)

James R. Brink, Ph.D.
Battelle Columbus Division
505 King Ave.
Columbus, Ohio 43201

Paul Storey
Sacramento Air Logistics Center
SM-ALC MMESD
McClellan AFB, CA 95652

ABSTRACT

EISE is an Air Force developed real-time computer network consisting of commercially available hardware and software components to support systems level integration, modifications and enhancements to weapons systems. The EISE approach offers substantial potential savings by eliminating unique support environments in favor of sharing common modules for the support of operational weapon systems.

An expert system is being developed that will help support diagnosing faults in this network. This is a multi-level, multi-expert diagnostic system which uses experiential knowledge relating symptoms to faults and also reasons from structural and functional models of the underlying physical model when experiential reasoning is inadequate. The individual expert systems are orchestrated by a supervisory reasoning controller, a meta-level reasoner which plans the sequence of reasoning steps to solve the given specific problem. The overall system, termed the Diagnostic Executive, accesses systems level performance checks and error reports, and issues remote test procedures to formulate and confirm fault hypotheses.

BACKGROUND

In general, once a weapon system has been operationally accepted and placed into the Air Force inventory, management responsibility for its support is transferred from the acquiring agency to one of the Air Logistics Centers (ALC) within the Air Force Logistics Command (AFLC). Aside from the classical logistics management functions, these centers also provide the engineering capability to do systems analysis, modifications of the hardware and software, component level testing and evaluation, and system level integration and test. The primary engineering tool for AFLC weapon system support is the Integration Support Facility (ISF).

A typical ISF, being a subset of the tools the contractor originally used to develop the system, is useful for supporting just the original system. Because modification is difficult, individual ISFs are developed to support the various models of the same weapon system. Like the weapon systems they

support, ISF support is costly, requiring thousands of personnel and hundreds of millions of dollars annually for the Air Force.

To alleviate these problems, the Air Force is developing the Extendable Integration Support Environment, or EISE. The EISE concept consists of common hardware and software modules for common integration support functions. These modules, or building blocks, are logically reconfigurable to provide support for multiple weapon systems within the same environment. The EISE building blocks are off-the-shelf, commercially available items to the greatest extent possible. The building blocks are connected by Ethernet for non-real time requirements, and by a high-speed token passing network during real time simulations. As a result, custom ISFs for each weapon system will no longer be needed, thus reducing support costs through resource sharing, economies of scale for sparing of the individual building blocks, and for facility maintenance contracts.

OVERVIEW OF DIAGNOSTICS APPROACHES

Diagnosing problems associated with accommodating a great variability in the building blocks for EISE on a real time high speed network is expected to be quite difficult. Thus, a complex expert system, named the "Diagnostic Executive", is being developed to ensure the functioning and availability of EISEs. It is an "executive" because it performs high level diagnostics and calls upon diagnostics in the various processors as required to identify and isolate faults in EISE. In order to provide background on the development strategy for the Diagnostic Executive, we first summarize existing approaches to diagnostic expert systems.

The conventional approach to diagnostic expert systems development involves collecting and organizing the knowledge gained through experience by repair technicians, essentially associating a set of symptoms to the set of faults causing those symptoms. This approach is termed surface, shallow, experiential or empirical reasoning and it works well where human maintenance experts have accumulated enough experience to provide rules of thumb for most of the probable faults. Early experiential diagnostic systems developed as a flat knowledge base with every rule being scanned at every step of inference. In spite of some weaknesses, this

approach has a proven track record and is largely responsible for the success the AI technology in the mid-1980's.

Life-cycle maintenance and evolution of flat rule based systems has proven to be quite difficult for systems of even modest size. For classification problem solving, this problem is substantially reduced by adding control to the inference process (called "establish-refine") and modularizing the rule base via a taxonomy of pre-enumerated fault conditions, organized in a top down hierarchy. The most likely probable fault class is "established" and taken as a "hypothesis" to be "refined" into more finely detailed probable fault classes in the next lower level of the hierarchy, and this process is repeated recursively. Separate sets of rules are used for each class to relate probable fault classes to supporting evidence and to summarize results.

Both experiential and fault classification systems are shallow approaches, and represent condensed and streamlined knowledge which can yield accurate results. However, if a condition arises which is beyond its expertise (i.e., the pre-encoded human diagnostic rules), the system will perform poorly. CSRL developed at the Ohio State University (1983, 1986) and commercialized by Battelle (1986) represents an excellent example of this approach to classificatory problem solving. Rule Kit by General Dynamics (1984) is another example of this approach.

Structural based troubleshooting approaches incorporate information about the topology of the Unit Under Test (UUT). This includes connectivity between each of the components, similar to the schematic diagram of the circuit used by human technicians. Given a known good input and a known bad output, the system can use the connectivity to trace signals through the diagram. This is more flexible and robust than the shallow approach because failures can be diagnosed without pre-entering symptom-fault relationships. On schematics with many layers of logic, many stages of processing between input and output, large fan-in or fan-out, the ambiguity groups can be large, and combinatorial explosion problems often exist. By including with each component its failure probability, cost of access, test setup costs, required testing times and information yield, techniques can be employed to optimize for least test cost, least time to locate, least technical skill required, or least test equipment required. Cantone (1984) and Simpson (1982) are examples of this approach.

The next advancement in diagnostic reasoning approaches is to model the behavior and functionality of the components in addition to the structure. Behavior of a component can often be described with a set of rules, however, these rules come from the design engineer. Once a device has been modeled, it is generic and can be used for multiple purposes. This "**model based reasoning**" approach is sometimes termed "deep reasoning" when reasoning from physical first principles.

The model based reasoning strategy for diagnosis involves noting the differences between the expected outputs as predicted by the model and reality as measured at test points. Connectivity is used to find the components along the topological path of the signal. The dimension of the discrepancies (in analog circuitry, frequency, amplitude, phase, etc.) is used to narrow the search to the components that affect that dimension of the signal. This is done by pattern matching amongst the behavior rules of each component. The further screening of possible failed components by functionality greatly reduces the ambiguity groups, and therefore speeds up the search and reduces the number of tests required to isolate the fault. The use of behavior/functionality to determine which components affect a given measurement is the most important contributor to the diagnostic power of the model based reasoning approaches. Conversion of quantitative test measurements to terms suitable for qualitative reasoning (lo, OK, hi, always, sometimes, never), propagation of constraints, conflict resolution via tracking of assumptions and dependency chains in a truth maintenance system are some of the techniques employed in the model-based reasoning systems.

Model based systems, unlike heuristic or classification systems, can be robust without requiring exhaustive a priori enumeration of symptom-fault relationships. Using the structure and function of the UUT, symptoms and effects of failures can be dynamically computed. The disadvantages are that development times are longer, and execution time can become computationally intensive. Genesereth (1982), Sembugamoorthy (1984), Davis (1984), Pipitone (1986), de Kleer (1986), and Kaplan, et al (1988) are good examples of the model based reasoning approach.

THE DIAGNOSTIC EXECUTIVE

The approach used in the EISE Diagnostic Executive employs a mixture of the approaches described above. Experiential knowledge consisting of a priori failure probabilities and heuristic rules of thumb are tried first to locate the most common failures. This gives a quick response for problems within its range of expertise. When the diagnostic system encounters a failure mode beyond its surface heuristic rules, the deeper or model based reasoning system takes over, using its knowledge of structure (connectivity) and functionality (behavioral models) to search for components whose failure can explain the given symptoms.

Others who have used this approach to building diagnostic systems include Fink and Lusth (1986), Richardson and Barthelminghi (1986), Chu (1988), Pau (1986), Havlicsek (1986), McCown and Conway (1988), Warn (1988).

The types of faults expected to create the most problems for deployed EISE systems fall into three categories. First, the hardware components of EISE can fail. Because EISE involves mostly off-the-shelf commercial hardware, isolating hardware failures need only be accomplished to the vendor responsible unit, which might be a workstation (e.g., Sun, MicroVax) or a card (e.g., Heurikon).

ORIGINAL PAGE IS OF POOR QUALITY

Second, timing problems can occur during the real-time simulation phase, causing problems ranging from bad data to total system shutdown. Isolating these problems is expected to be especially difficult because they can arise from a variety of sources including hardware failures (intermittent or otherwise), errors in the network software, errors in the applications software or configuration errors. Third, faults can result from an incorrect setup or operation of the simulation itself. These faults can occur when an operator does not correctly follow the setup protocols, when an object code file does not get properly downloaded to a processor, or when the incorrect, or old version of object code is downloaded to a processor.

Many of these problems are highly interrelated and may have unpredictable side effects. Error messages resulting when an anomaly finally surfaces and becomes detectable by system checks may be highly unrelated to the cause of the problem. Thus, the diagnostic executive incorporates reasoning approaches to resolve the resulting ambiguities.

IMPLEMENTATION

The Diagnostic Executive is currently being implemented on a Symbolics 3675 using Intellicorp's suite of knowledge engineering tools. Diagnostics information from each layer of the ESIE network (see Figure 1) is routed to the Diagnostic Executive, which calls the appropriate diagnostic routines and repair actions. The planned structure of the Diagnostic Executive (shown in FIGURE 2.) is a multi-level, multi-expert diagnostic system which uses experiential knowledge relating symptoms to faults and also reasons from structural and functional models of the underlying system. The individual expert systems, termed Reasoners, are orchestrated by a supervisor termed the Reasoning Controller, a meta-level reasoner which plans the sequence of reasoning steps to solve the given specific problem. The Reasoners are integrated via highly structured common working memory managed by the truth maintenance system which keeps track of all relevant facts, deductions, hypotheses and chains of reasoning. A conclusion reached by any reasoner serves to constrain the space of possible causes of difficulty known as the ambiguity group. The constraints from all reasoners are summed in the Ambiguity Group Truth Maintenance System as they are determined. This stepwise summation of constraints, known as propagation of constraints, has the tremendous advantage of limiting the size of search space. The principle is that two simple constraints by separate reasoners can synergistically add to tremendously reduce the size of the search space a third reasoner must look through.

Coordinating the multiple expert systems employed in the Diagnostic Executive is the responsibility of the "Reasoning Controller", a knowledge base that contains information concerning which reasoning strategy is best to employ for each type of diagnostic problem. Upon malfunction, the Reasoning Controller is activated to determine the state of the network, what parts are functioning correctly, and the nature and extent of the problem.

The Reasoning Controller consists of a planner, agenda, scheduler and progress monitor. Using the refinement of skeletal plans technique, the planner matches relevant features of the problem, symptoms, and states of the network or operator requests, and chooses a sequence of applying the Reasoners that best fits the problem at hand. The sequence is placed in the agenda and executed by the scheduler. The progress monitor is a regularly executed watchdog process which reports information on the current known state of all nodes, processors and functions as available from the current ambiguity group, its current strategy, goals and deductions. In the future, it is expected to compare current progress against established norms and time constraints so that replanning can be directed to the planner when required. Explanations are also available in the form of dependency records and tracings of rule firings.

The typical control strategy is for the Reasoning Controller to first invoke the Event Reasoner to look at current system status data (including reported status of each node and error conditions), and to trace the operator command input history, forward chaining from this information to deduce the estimated status of all processors on the network. Next, the Reasoning Controller invokes the Surface Reasoner, to identify probable fault classes that can explain the observed symptoms (using its experiential knowledge). Assuming the fault is not isolated by the Surface Reasoner, the Structural Reasoner is called upon to locate optimal test points in the system, given the malfunctions currently under diagnosis. The test points are chosen both to reduce the number of tests required to isolate a fault and to minimize the cost of performing the test. At this time, the Functional Reasoner can be used to determine expected values of intermediate test points from known good points or known bad points. Differences between the model predictions and the actual responses of the system constitute the symptoms to be used by the model based Reasoners.

The Event Reasoner is the primary diagnostic aid in beginning failure analysis of network startup from power-off condition to full-up, real-time condition. Major event sequences modeled include initialization of the individual computers, communication over a non-real-time network, the downloading of configuration files from servers to the diskless systems, communication over the real-time network, initialization of processors with real-time application data, handshaking and communication in real-time and post simulation activities.

The Event Reasoner uses models of temporal sequences to constrain the search to those portions of the system active during each action. Correct sequences of operations indicate processors and functions which have operated correctly. Thus the recognition of temporal sequences can serve as a landmark to indicate state of the system. The results of time sequences trigger rules to insert facts into the structural and functional dimensions of our ambiguity group truth maintenance system.

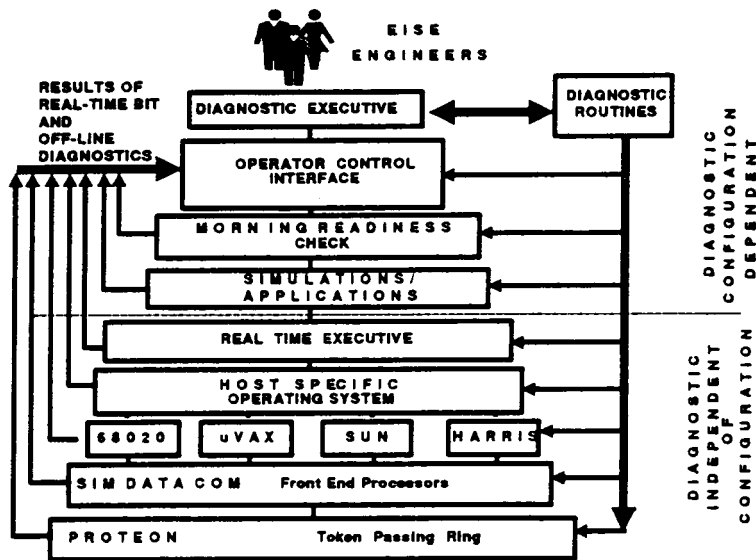


Figure 1. Overview of the Diagnostic Executive

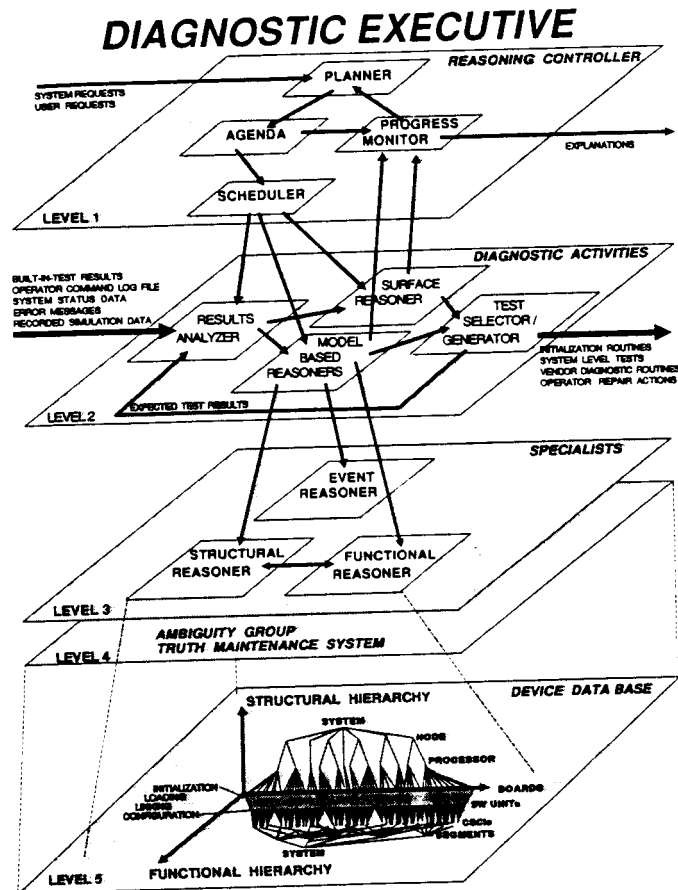


Figure 2. Architecture of the Diagnostic Executive

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

The Structural Reasoner is a topological based expert system as described in the Diagnostics Overview section. Each system component is represented as a unit, containing slots for source and destination of data paths. A connectivity tracer uses this information to find the paths from known good test points to known bad test points. After the paths have been found, feedback loops are marked to be opened, and calculation is done to find the point in the path that will yield the most information for the least cost. Included with each component is its failure probability, cost to access, test setup costs, required testing times and information yield. A computation is performed to optimize for least test cost, least time to locate, least technical skill required, or least test equipment required. The criteria to be optimized is passed down from the Reasoning Controller each time the Structural Reasoner is invoked. The result of this process is the optimal test point.

The Functional Reasoner is a model based reasoner as described in the Diagnostics Overview section. It models the transformations which occur as a signal is passed through the component. We have adapted the methodology of Pipitone (1986). A separate rule is used for every dimension of the signal. Qualitative reasoning (always, sometimes, never, low, OK, high) is used, not quantitative, numeric reasoning. All functionality rules are bi-directional. Computation of expected signal downstream from a known signal is done by forward chaining through any rules (always, sometimes, or never). Backward chaining can also be done from a measured failed testpoint to find components responsible for the erroneous behavior.

After the expected values have been computed, the test is run. Comparison of actual test results with expected test results yields a symptom. The dimensions of the symptom which differ from the predicted value is used by the Functional Reasoner to search for rules of each component along the test path that influence this dimension of the signal. This is done by matching on the functionality rules. Components along the signal path that contain rules that influence the dimension of the signal differing from prediction are the ambiguity set, the set of possible causes of the abnormality. Most of the diagnostic power of the Functional Reasoner comes from chaining on rules describing behavior. This narrows down the topological search to only those components that affect the behavior of the specific parameter under measurement.

The Ambiguity Group Truth Maintenance System constitutes Level 4 of Figure 2 and overlays the device data base. The deductions and corresponding rule firings are recorded in a set of worlds, or state/dependency graphs, managed by the truth maintenance system. Most of the intermediary conclusions made by the system are stored in these situation graphs or worlds. The truth maintenance systems adds, deletes, merges or invalidates worlds as information is confirmed, contradictions are found and new hypotheses are generated. Rule firings explicitly control this process. Facts not fitting

into the worlds structure are stored in an unstructured facts list.

The Device Data Base constitutes Level 5 of Figure 2. This data base contains a structural hierarchy of EISE, from system to node, to individual processors on the node, to cards inside each processor. Boards are the replaceable unit in the EISE system, therefore the structural hierarchy only goes down to the board level. The device data base also contains a functional hierarchy of EISE.

As indicated above, the architecture described here represents our implementation plan. Implementation began early in 1988, and a working version of the Diagnostic Executive for the A-10 EISE will be ready for validation in late 1988. Currently, the Event Reasoner has received the most implementation attention and can handle many of the problems encountered. Portions of the other reasoners have been implemented, but integration of all reasoners as described through the reasoning controller is as yet incomplete.

CONCLUSIONS

The diagnostic executive described here is being built on the A-10 EISE, i.e., the EISE which serves as the Integration Support Facility for the A-10. Other Air Logistics Centers are expected to use EISEs to support their weapon systems. Extending EISE to other weapon systems will not only introduce other configurations and hardware components to diagnose, but operator setup protocols are expected to be more complex and timing problems will be more severe because of greater traffic on the network.

The Diagnostics Executive will substantially decrease the need to employ many high cost troubleshooting experts. In addition, faster and more thorough diagnostics will decrease downtime allowing greater utilization of existing valuable computer network resources. A third important benefit will be increased availability of the Integrated Support Facility allowing faster turnaround time to implement the needed, timely and highly responsive updates to our aircraft systems.

ACKNOWLEDGEMENTS

Mr. Steve Rinehart and Mr. Michael DeVaney of Battelle are performing the detail design and implementation of the Diagnostic Executive. Most of the domain expertise is being supplied by the TRW Electronic Systems Group in Sacramento.

REFERENCES

- Bylander, T. and Mittal, S., "CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling, AI MAGAZINE 7(3):66-77, 1986.
- Cantone, Lander, & Gaynor. "IN-ATE/2: Interpreting High Level Fault Modes," in IEEE AUTOTESTCON, 1984.
- Chandrasekaran, B., "Towards a Taxonomy of Problem Solving Types," AI MAGAZINE 4(1):9-17, 1983.

, S., "Approaches to Automatic Fault Diagnosis: A Critical Evaluation", AI IN ARMNAMENT WORKSHOP, American Institute of Aeronautics and Astronautics, 1988.

Davis, R., "Diagnostic Reasoning Based on Structure and Function", ARTIFICIAL INTELLIGENCE, Vol. 24, 1984.

de Kleer, J. & B. Williams, "Reasoning About Multiple Faults", AAAI 86 PROCEEDINGS, American Association of Artificial Intelligence, 1986.

Dudzinski, E., J. Brink, and D. Sharma, "CSRL-From Laboratory to Industry", EXPERT SYSTEMS IN GOVERNMENT SYMPOSIUM, 1986.

Fink, P. and J. Lusth, "A Second Generation Expert System for Diagnosis and Repair of Mechanical and Electrical Devices", in AI APPLICATIONS FOR INTEGRATED DIAGNOSTICS, University of Colorado, 1986.

General Dynamics Electronics Division, "Rule Kit: A Set of Tools for Building Rule Based Diagnostic Systems," 1984.

Genesereth, M., "Diagnosis Using Hierarchical Design Models", Proceedings of National Conference on AI, AAI, August, 1982.

Havlicsek, B., "A Knowledge Based Diagnostic System for Automatic Test Equipment", Artificial Intelligence in Maintenance, University of Colorado, 1985.

Kalpin, S., R. Schrag, and L. Volovik, "Performing Electronic Diagnostics With Distributed Expert Systems", AI in Armament, American Institute of Aeronautics and Astronautics, 1988.

McCown, P. and T. Conway, "APU Maid: An Event-Based Model for Diagnosis", AI in Armament Workshop, American Institute of Aeronautics and Astronautics, 1988.

Pau, L. F., "Survey of Expert Systems for Fault Detection", Test Generation and Maintenance, EXPERT SYSTEMS, Vol. 3, No. 2, April, 1986.

Pipitone, J., "The FIS Electronics Troubleshooting System", in Computer, 19 (7), pp. 68-76, Copyright 1986 by Institute of Electrical and Electronic Engineers, 1986.

Richardson, J. and G. Barthelenghi, "AI in Test Program Development, in AI Applications for Integrated Diagnostics", University of Colorado, 1986.

Sembugamoorthy, V. and Chandrasekaran, B., "Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems," COGNITIVE SCIENCE, August, 1984.

Simpson, W. and H. Balaban, "The ARINC Research System Testability and Maintenance Program (STAMP)", IEEE AUTOTESTCON '82, Dayton, OH, 1982.

Warn, K., "Deep Reasoning Expert System for Armament Diagnostics Applications, AI in Armament Workshop", American Institute of Aeronautics and Astronautics,

AUTOMATIC DETECTION OF ELECTRIC POWER TROUBLES

(ADEPT)

Caroline Wang, Hugh Zeanah, Audie Anderson, Clint Patrick
Information and Electronic Systems Laboratory
Marshall Space Flight Center, NASA
Huntsville, Alabama

Mike Brady and Donnie Ford
University of Alabama in Huntsville

ABSTRACT

ADEPT is an expert system that integrates knowledge from three different suppliers to offer an advanced fault-detection system, and is designed for two modes of operation: real-time fault isolation and simulated modeling

Real time fault isolation of components is accomplished on a power system breadboard through the Fault Isolation Expert System (FIES II) interface with a rule system developed in-house. Faults are quickly detected and displayed and the rules and chain of reasoning optionally provided on a Laser printer.

This system consists of a simulated Space Station power module using direct-current power supplies for Solar arrays on three power busses. For tests of the system's ability to locate faults inserted via switches, loads are configured by an INTEL microcomputer and the Symbolics artificial intelligence development system. As these loads are resistive in nature, Ohm's Law is used as the basis for rules by which faults are located.

The three-bus system can correct faults automatically where there is a surplus of power available on any of the three busses. Techniques developed and used can be applied readily to other control systems requiring rapid intelligent decisions.

Simulated modeling, used for theoretical studies, is implemented using a modified version of Kennedy Space Center's KATE (Knowledge-Based Automatic Test Equipment), FIES II windowing, and an ADEPT knowledge base. A load scheduler and a fault recovery system are currently under development to support both modes of operation.

INTRODUCTION

Marshall Space Flight Center (MSFC) is involved in design and development of the Automation of Electrical Power Systems project. This demonstrates the feasibility of using computer software to enhance fault-diagnosis techniques and develop fault-recovery techniques for the Space Station. To accomplish this, prototype software was developed to automate such tasks as detecting and isolating faults and monitoring and reasoning status.

The ADEPT system includes:

- (1). Real time fault isolation through a breadboard modeling the power components.
- (2). A local simulator which uses the theoretical models and will eventually support the fault recovery system.

HISTORY BACKGROUND

In 1985, Martin Marietta Denver Aerospace delivered to MSFC the Fault Isolation Expert System including a two-rack, 350-watt, three-channel electrical power system breadboard.

MSFC was experimenting with various software techniques to improve performance and speed. ADEPT was built with the MSFC rule system utilizing the existing FIES II breadboard and software interface and KATE as a tool for the local simulator.

The real-time fault isolation version was implemented in LISP, because of its ability to search for a fault, display fault data, and automatically print out the fault reasons and current data along with the steady-state data for comparison.

The University of Alabama in Huntsville is also involved in this project. They have already converted the software into Symbolics system genera 7.1, and also will be conducting a future study of load management and scheduling.

THE ADEPT SYSTEM

ADEPT is composed of a Symbolics 3670 computer linked to the modified FIES II system. The Symbolics 3670 includes a high-resolution graphics terminal, eight megabytes of memory, a 474 megabyte hard disk, Laser graphics printer, and a LISP environment. The FIES II breadboard is built into two side-by-side racks containing the host computer, its memory storage devices and I/O support equipment; the relay board subrack, its power supplies and related controllers; communications boards, ports, and cables; housekeeping power supplies; control switches and lighted displays.

Figure 4 outlines the components in the ADEPT system and the interactions of these components with one another.

Data transfer scheduling and control are provided by the host computer, an Intel System 86/380. Based on the iRMX86 operating system, the 86/380 contains the iSBC 86/30 Single Board Computer board, a thirty-five megabyte Winchester hard-disk, a one megabyte eight-inch flexible disk drive, and a multibus expansion rack with slots containing not only controllers for the computer itself, but also communication and data conversion boards discussed below. Software run on the 86/380 is written in Intel's ASM86 assembly language.

Three dual-sided power supplies provide charge to the batteries or electricity to drive the system's load resistances, or both, depending upon the configuration into which the busses' relays are set. Representing the Space Station's solar arrays, these supplies are capable of up to fifty volts and nearly two amperes output on each of the six available channels. Each supply has independent current-limiting adjustment, allowing simulation of various solar array lighting conditions.

At the heart of the breadboard is the relay board subrack, comprised of six boards containing forty-eight relays along with related support components. In addition to its function as the system's switching center, the relay subrack provides attach points for most of the sensor lines, by which A/D converters sample system conditions, and all of the fault insertion lines. The fault insertion logic, used to introduce various abnormalities at nodes along the power busses, sends its outputs directly to the relay boards where the "support components" mentioned effect conditions of open or closed relay faults and resistive or direct shunt faults. Configurations may be inserted either manually, using toggle switches on the front panel of the FIES II racks, or remotely, from the Symbolics terminal or the debugging monitor. In the event that a switch is offset from the normal mode, the corresponding relay cannot be controlled remotely, but a fault will exist and should be detected and isolated.

The system integrates software from three different suppliers to offer an advanced fault detection system, designed for the two modes of operation outlined in Figure 1.

Real-time fault isolation of components on the power system breadboard through the FIES II interface is made possible with the MSFC rule system. Faults are quickly detected, displayed, and reasoning provided. The FIES II interface and MSFC rules system were written in Common Lisp (Figure 2).

The simulation version uses a frame-based system, describing all system components and the relationships among them. A constraint system analyses these relationships and compares theoretical to actual measured values, thus identifying constraint failures (Figure 3).

FAULT ISOLATION

When an initial configuration of loads is selected and downloaded from the Host Computer, and steady-state condition is achieved, all the sensor points' voltages and currents are read continuously and any significant change at any sensor point indicates a fault has been inserted. This Fault condition is then flagged to the Host computer to initiate the isolation program.

Fault Type: OPEN RELAY

Open circuit conditions are indicated by a sudden drop in the values of current read at the sensors while the voltage values remain the same or perhaps higher. Isolation is done by searching for a sensor point where the voltage is zero. The location of the inserted fault lies between the sensor points where there was a voltage and was not a voltage.

Fault Type: DIRECT SHUNT

A direct shunt, or short-circuit, fault causes a sudden increase in the sensor readings of current values and a decrease in voltage values on sensors nearest the power source. When this occurs, the fault type is identified and a search begins for a sensor point where the current is higher than the steady state current and following points have current readings of approximately zero.

Fault Type: RESISTIVE SHUNT

Resistive shunt causes a sudden increase in current readings on the sensors nearest the power source. A decrease in the voltage may also occur where the load plus the resistive shunt causes the current to exceed the capacity of the solar cells being simulated. Isolation of the resistor shunt fault is done by

ORIGINAL PAGE IS OF POOR QUALITY

REFINEMENTS

identifying the first sensor reading with a significant decrease in current. The fault is between this sensor and the last one back toward the power source with a high current reading.

Refinements in the rules are made using Ohm's Law to further identify the type of fault being experienced. This is done by considering the ratio of the values of currents and voltages between steady-state and fault conditions.

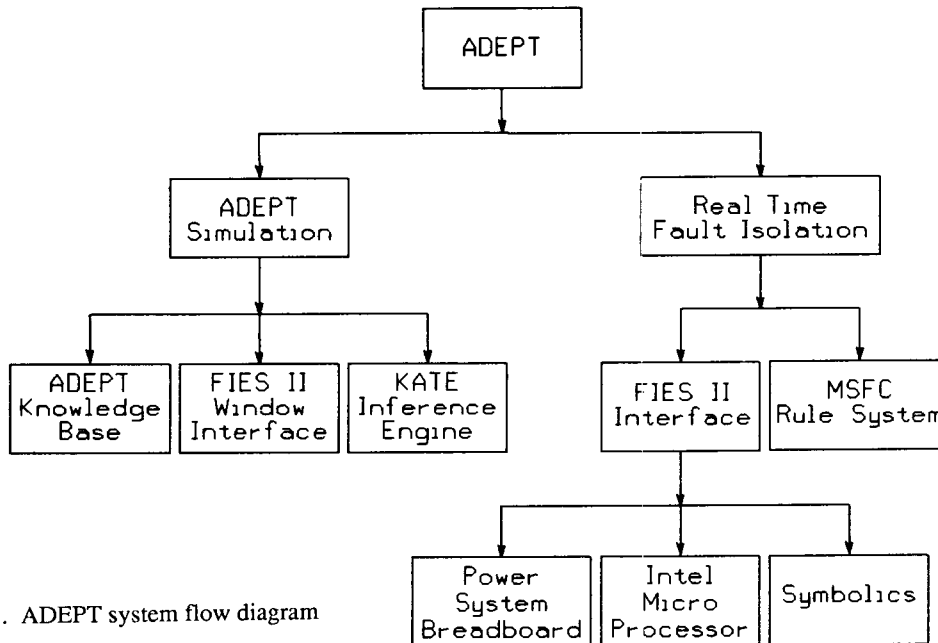


Figure 1. ADEPT system flow diagram

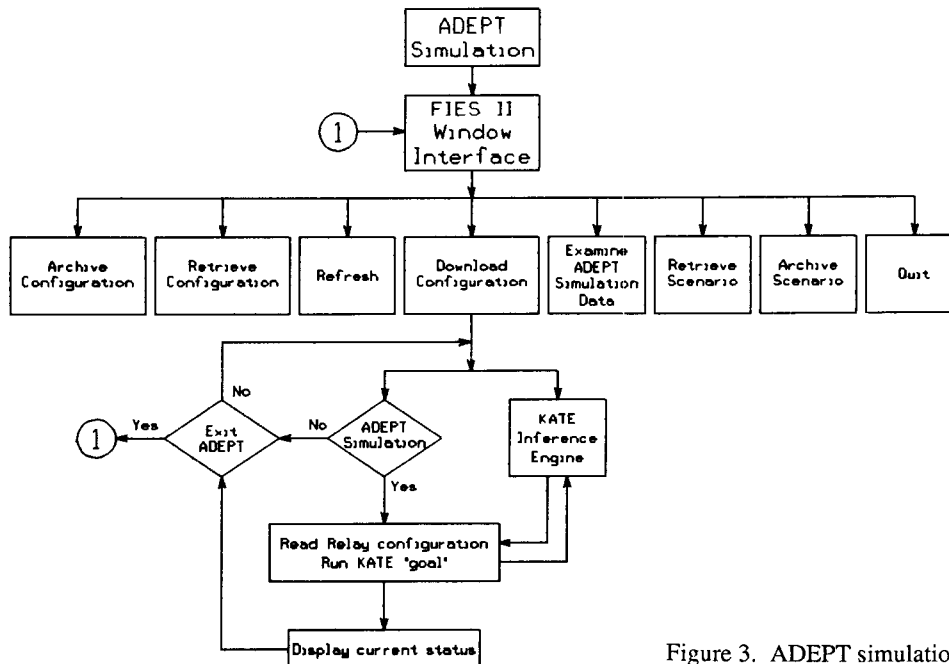


Figure 3. ADEPT simulation flow diagram

ORIGINAL PAGE IS
OF POOR QUALITY

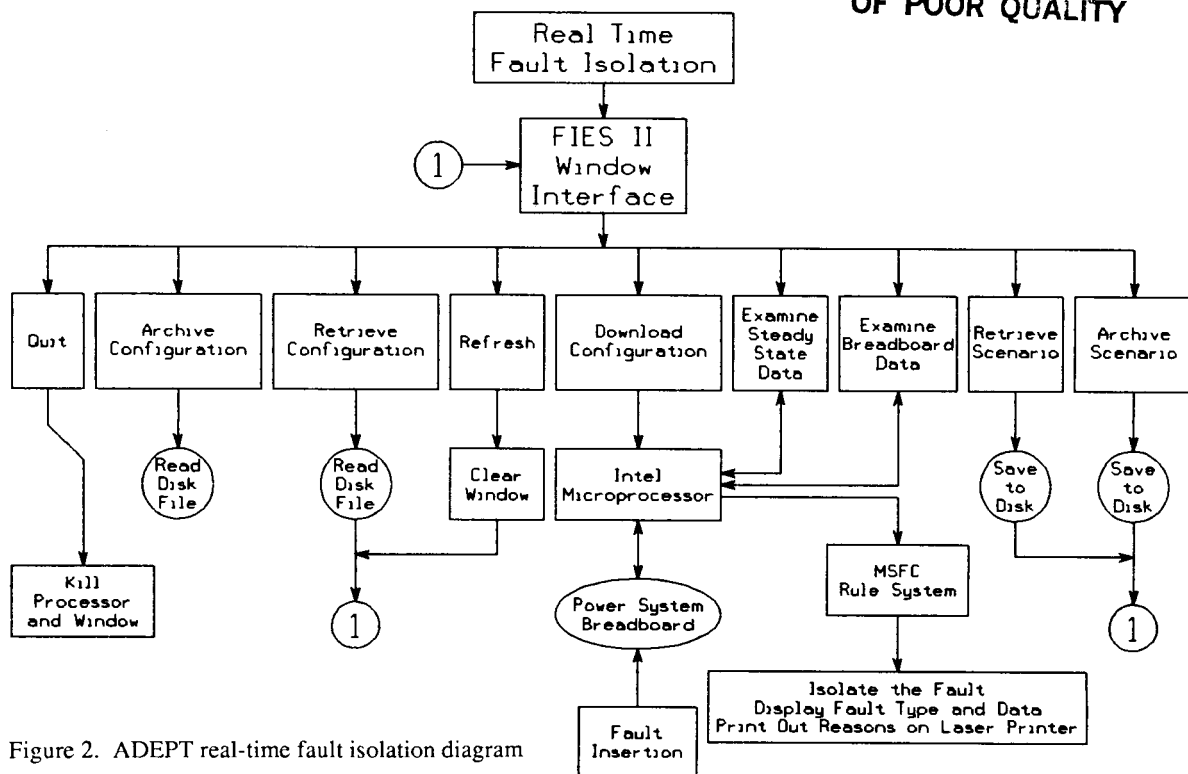


Figure 2. ADEPT real-time fault isolation diagram

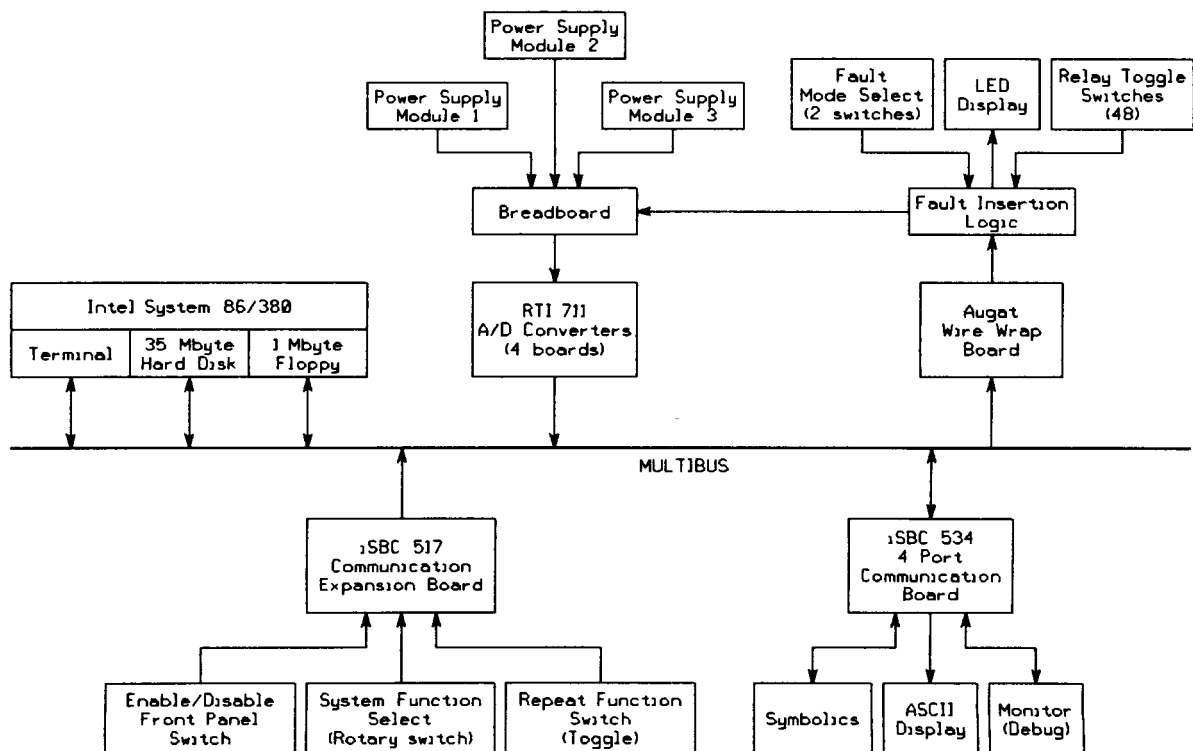


Figure 4. Hardware flow diagram

REAL TIME EXPERT SYSTEM PROTOTYPE
FOR SHUTTLE MISSION CONTROL

John F. Muratore
Robin M. Madison
Troy A. Heindel
Terri B. Murphy
NASA Johnson Space Center

Robert F. McFarland
UNISYS Corporation

Arthur N. Rasmussen
The MITRE Corporation

(Paper not provided by publication date.)

Knowledge-Based Operation and Management
of Communications Systems*

Harold M. Heggstad
M.I.T. Lincoln Laboratory
244 Wood Street
Lexington, MA 02173

ABSTRACT

Expert systems techniques are being applied in operation and control of the Defense Communications System (DCS), which has the mission of providing reliable worldwide voice, data and message services for U.S. forces and commands. Thousands of personnel operate DCS facilities, and many of their functions match the classical expert system scenario: complex, skill-intensive environments with a full spectrum of problems in training and retention, cost containment, modernization, and so on. Two of these functions have been the subject of research programs at Lincoln Laboratory over the past two years, sponsored by Rome Air Development Center and the Defense Communications Agency respectively, namely 1) fault isolation and restoral of dedicated circuits at Tech Control Centers and 2) network management for the Defense Switched Network (the modernized dial-up voice system currently replacing AUTOVON). An expert system for the first of these is deployed for evaluation purposes at Andrews Air Force Base, and plans are being made for procurement of operational systems. In the second area, knowledge obtained with a sophisticated simulator is being embedded in an expert system. The background, design and status of both projects will be described.

1. **INTRODUCTION**

In order to maintain peak performance despite the fact that no electronic equipment can run indefinitely without degradation or failure, all communication systems must provide for detecting and correcting deficient operation. The growing technical disciplines of Network Management and of "AO&M" (Administration, Operation and Maintenance) in the telecommunications industry reflect the substantial payoffs that can be obtained by prompt and careful attention to these factors, keeping network performance and revenues

continually close to peak design capabilities. A number of efforts have been undertaken to provide automated aids for human operators carrying out these functions, and in recent years artificial intelligence techniques have been pursued in the attempt to achieve consistently high performance despite operator skill and experience limitations [1].

For military communication systems, the motivations for continually maintaining high network performance are slightly different. For one thing, chronically tight defense budgets tend to limit communications expenditures to the bare minimum, and adequate support of military requirements can only be achieved if these minimum systems can be kept tuned to their peak capability. Another significant difference is that military communication systems are precedence-oriented: in times of emergency or network damage the best achievable service must be provided to the most critical users, even if this requires preemption or denial of service to less essential users. Moreover, military operators and technicians tend to be young and inexperienced compared to their civilian counterparts; this increases the risk that military networks may not be at their best, and creates even greater need for automated aid systems.

The purpose of this paper is to briefly describe two ongoing projects which are developing Expert Systems techniques for assisting military personnel in maintaining peak performance of military voice and data communications systems [2,3,4,5]. One project addresses Technical Control, which is the process of isolating faults and restoring service on critical dedicated circuits. The other project addresses Network Management for the worldwide voice system called the Defense Switched Network; this is the process of

allocating and controlling network resources to assure reliable service for high-precedence dial-up users, despite failures or congestion in major portions of the network.

2. THE EXPERT TECH CONTROLLER

Work has been in progress since FY86 on the Expert Tech Controller (or ETC), an expert system for assisting humans in performing circuit fault isolation and service restoral at military Tech Control Facilities (TCFs) [2,3,4]. A worldwide network of some 400 TCFs performs these functions for more than 61,000 dedicated circuits operated by the Department of Defense (DoD) for facilities or users who must have full-time connectivity because they are continually active, or because their mission requires instant communications in the event of emergency. Such circuits include, for example, high-usage long-haul trunks for the military switched voice network; dedicated circuits linking the Pentagon and the White House with military force commanders; and data links joining various essential computer systems.

ETC was initially implemented with ART (trademarked acronym for Automated Reasoning Tool, an Expert System shell produced by Inference Corp.), on a Symbolics 3640 computer. This approach yielded precisely the advantages that one hopes for under such conditions: we were able to rapidly prototype a system that performed circuit fault isolation, focusing our attention on acquisition and understanding of knowledge in the problem domain rather than expending energy on the writing of software tools and facilities. By being able to frequently come back to the domain knowledge sources (senior NCOs at a major TCF, the 2045th Communications Group at Andrews Air Force Base, Washington, DC) with working software implementations of the knowledge they had given us in our previous visit, we were able to sustain a high level of enthusiasm and cooperation.

As ETC's knowledge base grew its operation became slower and slower, however, especially during resets. Our analysis indicated that the generality and power of the ART design brought along much overhead that was not needed for the particular kinds of problems ETC had to deal with. Accordingly, we decided to reimplement the system entirely in ZetalISP, the native language of the Symbolics machine. The resulting performance was entirely satisfactory; ETC moved along at about the rate at which a skilled human operator would reason about the problem in hand. Development and extension of the system

knowledge base continued steadily thereafter until about mid-FY88, when it was decided to suspend further development of ETC and transfer our efforts entirely to an advanced project called MITEC (Machine-Intelligent Tech Controller), described below.

3. ETC FUNCTIONALITY

1,000 or more circuits pass through a typical TCF, where the Tech Controllers can access them for test, patch and re-route purposes via banks of manual patch panels. Having been gradually implemented over a number of years, these circuits involve a potpourri of equipment types and vintages, and the acquisition of fault isolation skills for all the commonly used variations can require years of practice.

The basic mission of Tech Controllers is to insure that the circuits passing through their TCF are continually available and operating at peak efficiency. Whenever a circuit outage occurs, Tech Controllers rapidly isolate the faulty segment and patch around it with spare facilities, or with facilities preempted from lower-precedence circuits, to restore service. At this point a repair order is issued for the appropriate repair service to find and fix the specific equipment failure. During a normal working day at the Andrews Air Force Base TCF there are typically several of these circuit outage problems in progress at once. On a less urgent basis, Tech Controllers endeavor to minimize failures by performing routine quality assurance testing on working circuits, in order to identify and correct incipient problems before they occur. The workload resulting from all these duties is substantial, and the actual personnel complement on board at a TCF is typically somewhat below the authorized level; moreover, two-thirds of these are likely to be trainees. Consequently there is great interest in the possible application of Expert Systems techniques for raising personnel work efficiency by allowing them to work at higher skill levels.

The initial objective for the ETC development was to create a concept demonstration model showing how an Expert System could perform in the TCF environment. The design goal for ETC was the capability to isolate the causes of the majority of the normal kinds of problems (e.g., no signal, receiving garble, excessive retransmissions) on the many types of circuits (e.g., voice, data, teletype, digital) using the wide variety of communication links (e.g., land lines, HF radio, microwave, satellite, etc.) between TCFs. The initial concept

assumed an "air gap" between the problem-solving logic and the communications equipment: ETC directed the measurement and data-gathering actions of a human operator via CRT displays, and the human supplied the requested information to ETC via mouse and keyboard. The problem-solving logic applied by ETC reflected knowledge obtained from skilled human practitioners. Graphics displays and text messages aided the novice operator in visualizing and understanding the fault isolation processes. Upon isolating the faulty circuit segment, ETC would supply instructions for the operator on how to patch around it. Finally ETC would complete the paperwork items required of human operators under normal circumstances, namely a log of the isolation and restoral procedures completed and a repair order for fixing the faulty equipment.

By early FY88 the indications were that the concept demonstration goals of ETC had indeed been achieved. It was estimated that ETC's knowledge base encompassed the circuit, device and fault types involved in more than half of the normal daily work load at Andrews. Planning was begun for a follow-on system development that would "close the air gap" by allowing ETC to directly access the communications and test equipment, find the fault, and electronically patch around it. This system would exploit modern remotely-controllable communications, access and test equipment typical of that in use in the commercial telecommunications industry, and gradually being installed in military facilities. This new system, called MITEC (Machine Intelligent Tech Controller), would be targeted for introduction in the field on the same time scale as the modern communications and test equipment.

At the present time, further development of ETC has been suspended and the design of MITEC is in progress. A communications testbed is being assembled to serve as a development and demonstration environment for MITEC. This testbed represents two modern TCFs joined by digital trunk circuits in the 24-channel industry standard 1.544 Mbps DS1 format (often referred to as a T1 carrier). Each TCF has a group of local users, and each is controlled by its own Expert System. The testbed includes voice and digital user terminal equipment; modems and telephone lines; first- and second-level multiplexers; a DACS-II cross-connect switch for T1 trunks; and HLI 3200 test access switches provided with HLI 3701 and 3705 test sets. In-service and spare circuits are provided at several levels,

and remotely-controlled matrix switches can select the desired configuration. The initial goals for MITEC are to demonstrate fault isolation and service restoral on all the circuit and trunk fault variations that are possible on the testbed, which will represent the majority of situations likely to be encountered at modernized TCFs. These processes will proceed with no air gap, that is, with no direct human interactions other than keeping the operator informed via screen messages, and giving the operator the go/no-go decision authority before a suggested circuit patch is actually executed.

4. SIMULATION AND EXPERT SYSTEMS FOR DSN NETWORK MANAGEMENT

The DSN (Defense Switched Network) is currently being implemented as a modern replacement for the AUTOVON (Automatic Voice Network) system that was originated in the 1950s to provide reliable voice service for military commanders in CONUS and overseas. The service concept in both cases is direct-dial long-distance service between authorized telephones on military installations, carried on government-leased or -owned trunk circuits (which are a major category, by the way, of the dedicated circuits handled by the Tech Control Facilities discussed above).

A key feature of the system is a five-level precedence and preemption structure (Routine, Priority, Immediate, Flash, and Flash Override), in which a call being placed by a higher-precedence user can automatically preempt lower-precedence calls in progress if necessary. Another feature of the system is that it is engineered to provide good service (i.e., low blocking probability) for precedence users at the lowest possible cost. Basically, this means that the number of expensive long-distance trunks between pairs of switching nodes is made as small as possible. Routine users, who generate at least two-thirds of the normal peacetime traffic, therefore get significantly higher blocking probability on AUTOVON/DSN than civilian customers experience on the commercial networks.

The baseline requirement for Network Management in this Spartan environment is to do the best possible job of providing non-blocking service to precedence users in the face of traffic overloads, equipment failures or other disrupting influences. In the antiquated AUTOVON system the provisions for network management were minimal; in some cases certain manual actions were possible (such as re-programming switches to block calls to a failed switch), but for the

most part the response to network problems was to dispatch repair crews and wait for service to be restored. A key feature of the DSN program is the replacement of the aging, limited AUTOVON switches with modern computer-controlled equipment that offers far greater Network Management power and flexibility. Two immediate problems arise in seeking to take advantage of this power: 1) there is no pre-existing body of DSN Network Management knowledge, and 2) it appears that the tasks of the DSN Network Manager will be complex and demanding, creating serious manpower training and retention needs.

The ongoing program in DSN Network Management at Lincoln Laboratory [5] has two main thrusts in addressing these problems: creating NM knowledge by experimentation with a powerful DSN simulation, and embedding this knowledge in an Expert System capable of advising less-skilled human operators as well as retaining a corporate memory of NM knowledge through personnel transfers and returns to civilian life.

5. THE CALL-BY-CALL SIMULATOR (CCSIM)

A large Fortran program has been developed which simulates all the DSN activities relevant to NM on a call-by-call basis. Its host computer is a Sun 3/260 work station, which typically simulates faster than real time (depending upon the size and complexity of the network being simulated). A CCSIM run is initialized with the topology and connectivity of the network under study, typically all the backbone switching nodes in a theater-wide DSN (i.e., Pacific or European), and is also provided with the matrix of average busy-hour routine and precedence traffic levels for each source/destination pair in the network. Random number generators initiate calls in accordance with statistical models of caller behavior, with averages matching the given matrices. Every event associated with each call is modelled, including all route selection processing, blocking and preemption events at source, destination and intermediate nodes, and user retry behavior. As the simulation progresses the experimenter can apply overload and fault conditions, and he can select and apply network management control actions from an available repertoire which, in the real world, would be transmitted to switch control computers throughout the network and would cause modifications in the way switches process subsequent calls.

CCSIM produces two classes of output information: concise local statistics reports for each network node, identical to the 5-minute

summaries continually transmitted to central authority by real switches in the field, and exhaustive reports of the details of the simulation run. The former constitute a set of "soda straw" views of the network that will be the only statistics information available to DSN network management personnel at theatre headquarters (the Area Communications Operations Center or ACOC), while the latter provides the experimenter with omniscient understanding of what really happened during the run. Such precise and complete information is obtainable because CCSIM actually tracks every simulated call through its complete history, from birth to death.

The process of Knowledge Engineering that is currently ongoing with CCSIM involves the initiation of specific damage or overload events during a run, followed by analysis of the switch reports to discern patterns and indicators that a network manager at the ACOC could have recognized as evidence of the existence of the particular fault condition. The experimenter then chooses a candidate NM control command and applies it to CCSIM, analyzing both the switch reports and the comprehensive statistics for indications that the control is successfully minimizing performance degradation caused by the fault condition. This knowledge engineering process is quite painstaking, involving many repetitious experiments to achieve statistical regularity as well as to understand the effects of parameter variations in both the damage and control commands. Preliminary results of this experimentation are described in [5].

In the near term, a separate effort has been undertaken to create an ACOC operator training system to develop personnel skills to meet the immediate needs of manual network management of the DSN, which is currently in the process of implementation. This training system will use the CCSIM to produce 5-minute reports as inputs to the existing operator console and support equipment. A training supervisor will set up CCSIM runs with representative fault conditions, and the operators will learn to recognize the statistical signs of trouble and to select and apply control actions, which will then be reflected in the ongoing operation of CCSIM.

6. THE NETWORK MANAGEMENT EXPERT SYSTEM (NMES)

An Expert System is being implemented to alleviate the DSN NM personnel training and retention problems. The long-range goal of this effort is to aid the ACOC NM personnel by performing pattern recognition on the

incoming stream of 5-minute switch reports to detect problem conditions, then to recommend NM control actions to overcome the problems -- in short, to embody and make available the store of NM knowledge developed through experimentation with CCSIM, and also to be augmented over time with the accumulated experience of the operations personnel. In the near term, the NMES is being integrated with CCSIM to form an interactive engineering tool in which the expert system can diagnose and correct problem conditions in the simulated network. This integrated system will be used in advancing the knowledge development for network management, as well as for other types of network engineering.

The Network Management Expert System has been implemented with ART, running in a Common LISP environment on a Sun 3/260 work station. The nature of the NM problem seems better matched to the ART structure than was the Tech Control environment, and it seems likely that the implementation will stay in ART for the time being. The NM problem is much more a matter of scanning collections of slowly-varying facts to see whether rules are satisfied, in contrast with the sequential nature of circuit fault isolation exercises. Moreover, while ETC had to be reset after every fault diagnosis in order to clear its world of all the schemata that were created while proceeding down the various unsuccessful and successful lines of inquiry, NMES can avoid time-consuming resets because, once it is turned on, it maintains an essentially continuous view of its problem domain.

A group of NMES software modules called "monitors" process the incoming switch reports from CCSIM, each watching for a particular pattern suggested by our knowledge engineering activity. An abstract state model of the network is maintained, including the nature and location of each of the problem indicators noted by the monitors. Higher-level modules analyze the network state, postulate problem conditions, and then confirm or reject the conclusions over successive 5-minute intervals. Another module consults the knowledge base of NM control actions to correct confirmed problems, and sends instructions to CCSIM to implement the controls at all the switch locations specified. An additional module watches the effects of the controls applied, both to determine whether the controls or parameters should be modified, and to remove the controls as soon as the problem condition goes away. Each knowledge module in the expert system has its own set of monitors that can be turned on or off, to scan the switch reports for patterns of interest.

7. SUMMARY

Military communications systems are subject to manpower skill, training and retention problems of a quite different order than their commercial counterparts, and are thus a fertile field for the development of knowledge-based software support systems. Moreover, commercial automated aid systems tend not to be applicable to the military problems, because of such differences as precedence and pre-emption capabilities. Two problem areas have been selected for concept validation development of expert system techniques addressing military needs, namely Technical Control and Network Management. Both systems have been developed to the point of substantial functionality, and appear likely to lead to transfer of the technology into the field.

REFERENCES

- [1] Liebowitz, Jay, "Expert Systems Applications to Telecommunications", Wiley, 1988.
- [2] Annual Report, "Knowledge-Based Systems Analysis and Control", MIT Lincoln Laboratory, 30 September 1986 [ESD-TR-87-041, AD A188 163]
- [3] Heggstad, H. M., "Dedicated Circuits Network Survivability Enhancement via Expert System Techniques", ICC '87, Seattle, WA, June 1987.
- [4] Otis, B. W., and Heggstad, H. M., "The Expert Tech Controller: A Network Control Expert System", MILCOM '87, Washington, DC, October 1987.
- [5] Annual Report, "Knowledge-Based System Analysis and Control-Defense Switched Network Task Areas", MIT Lincoln Laboratory, 30 September 1987 [ESD-TR-87-268, AD (not yet assigned)].

*This work was sponsored by the Defense Communications Agency and the Department of the Air Force.

The views expressed are those of the author and do not reflect the official policy or position of the U.S. Government.

NESSUS/EXPERT: BRIDGING THE GAP BETWEEN
ARTIFICIAL INTELLIGENCE AND FORTRAN

Pamela K. Fink, Ph.D.
Karol K. Palmer

Southwest Research Institute
6220 Culebra Rd.
San Antonio, Texas 78284
(512) 522-3288

ABSTRACT

The development of a probabilistic structural analysis methodology (PSAM) is underway at Southwest Research Institute (SwRI) as part of a research program for NASA/Lewis. In the near-term, the methodology will be applied to designing critical components of the next generation space shuttle main engine. In the long-term, PSAM will be applied very broadly, providing designers with a new technology for more effective design of structures whose character and performance are significantly affected by random variables.

The software under development to implement the ideas developed in PSAM resembles, in many ways, conventional deterministic structural analysis code. However, several additional capabilities regarding the probabilistic analysis makes the input data requirements and the resulting output even more complex. As a result, an intelligent front- and back-end to the code is being developed to assist the design engineer in providing the input data in a correct and appropriate manner. The type of knowledge that this entails is, in general, heuristically-based, allowing the fairly well-understood technology of production rules to apply with little difficulty. However, the PSAM code, called NESSUS, is written in FORTRAN-77 and runs on a DEC VAX. Thus, the associated expert system, called NESSUS/EXPERT, must run on a DEC VAX as well, and integrate effectively and efficiently with the existing FORTRAN code. This paper discusses the process undergone to select a suitable tool, identify an appropriate division between the functions that should be performed in FORTRAN and those that should be performed by production rules, and how integration of the conventional and AI technologies was achieved.

1 INTRODUCTION

1.2 Background To The Problem

Structural analysis techniques traditionally have been based on deterministic methods. That is, design characteristics such as geometry, material properties and loads were assumed to be constant. In reality however, there are factors such as manufacturing processes and operating conditions which introduce variance into design characteristics. In the past, this variance has either been ignored or a worst-case scenario has been adopted. Ignoring the variance may lead to design failure, the cost of which may be very great, perhaps involving the loss of human life. Adopting a worst-case scenario typically yields an extremely conservative and expensive design.

To account for the variability in design, structural analysis methods must step beyond the limitations of deterministic methods. The use of probabilistic structural analysis methods yields the ability for the designer to identify the relationships between specific design features and risk of structural failure. This provides information for educated decisions concerning risk, cost and need.

1.2 A Probabilistic Approach to Structural Analysis

A probabilistic structural analysis methodology (PSAM) is being developed at Southwest Research Institute (SwRI) as part of a research program for the National Aeronautics and Space Administration (NASA). In the near-term, the methodology will be applied to design critical components of the next generation space shuttle main engine. In the long-term, PSAM will be applied very broadly, providing designers with a new technology for more effective design of structures whose character and performance are significantly affected by random variables.

The objective of PSAM is to establish a computer-based methodology for design modeling of the real world effects of variability in applied loading (pressure, temperature, centrifugal force, etc.), material characteristics, tolerances, fits (boundary conditions), and strength data. The resulting computer programs will enable the design engineer to model how these design input variations (random variables) lead to variations in structural performance - stress, buckling, load, vibration amplitude, etc. The risk of structural failure can then be calculated by comparing the variation in performance with the variation in the allowable design limit condition, based on experience and material properties.

One goal of this NASA effort is to extend the research on probabilistic structural analysis methods and to develop a set of FORTRAN modules that embody the results of this research. The resulting system, called NESSUS (Numerical Evaluation of Stochastic Structures Under Stress) will include a module which performs standard deterministic structural analysis, a module which performs probabilistic analysis, a preprocessing module which will convert raw data to information that is usable by the probabilistic module, and a module which will perform a preliminary analysis of the results of the probabilistic module.

ORIGINAL PAGE IS
OF POOR QUALITY

1.3 Where Artificial Intelligence Is Needed

The ultimate goal of the project is to go a step beyond building a set of analysis modules to creating an integrated, user-friendly structural design package. However, the NESSUS structural analysis modules alone constitute only a set of state-of-the-art structural analysis programs. To upgrade these modules to a design package they must be integrated into one system and a user interface must be added. Without the interface the design engineer must proceed through a time consuming and complex set of steps to use the NESSUS modules. The engineer must first build a data deck which contains a description of the finite element model of the structure to be analyzed. Though this data deck may consist of several thousand lines of data and must be arranged in a specific format for the NESSUS modules, the type of information it contains is generally required by any conventional structural analysis program. The next step, involving probabilistic analysis, however, will be new to most design engineers. These data requirements for NESSUS are very challenging, generally far exceeding that needed for the conventional, deterministic design approach. Once the deterministic and probabilistic portions of the data deck have been built, the engineer must execute the appropriate NESSUS module(s). The output from a NESSUS run consists of a file full of numbers which must be interpreted and analyzed. Finally, the engineer must determine, from the results, what step to take next.

These are complex tasks which require expertise not only in finite element modeling, but also in the use of both the conventional and probabilistic NESSUS analysis modules. This expertise, however, is scarce since most engineers do not have experience with probabilistic analysis and even fewer have experience with the NESSUS modules. In addition to being complex, these tasks can be very time consuming. The analysis programs tend to run on large, number-crunching machines, such as the Cray, and can take many hours to complete a single run. Thus, a considerable amount of engineering time could be spent debugging erroneous data decks. These are classic reasons for using an expert system approach. An additional argument for using an expert system for this effort is that NESSUS is an evolving application, thus any interface to NESSUS must be flexible, expandable and maintainable. These are attributes which a rule-based expert system would provide. Therefore, it was decided to investigate the use of an expert system to serve as an aid to the design engineer in creating an appropriate and correct data deck and in analyzing the results of an analysis run. This expert system is called NESSUS/EXPERT.

2 IDENTIFYING THE REQUIREMENTS

2.1 Functional Requirements

NESSUS/EXPERT must serve as a flexible, user-friendly, integrated interface to the NESSUS structural analysis modules. In support of this function NESSUS/EXPERT must 1) be menu-driven and present to the user only those activities which are appropriate given the current status of the system, 2) invoke the various NESSUS modules directly at the appropriate times, 3) allow the user to leave the system at any stage with no loss of information, and 4) cater to users with varying degrees of expertise. In addition to these general system-wide requirements, the functional requirements of NESSUS/EXPERT can be categorized into two major functions: an intelligent user interface front-end and an intelligent back-end analysis aid.

2.1.1 An Intelligent Front-End to NESSUS

The front-end to the NESSUS structural analysis modules essentially provides an enhanced, on-line, automated user's manual. The NESSUS modules expect as input a data deck containing all of the structural information in a specific format. For large jobs this data deck may contain several thousand lines of data. To ease the chore of entering this data, NESSUS/EXPERT must infer data where possible and use defaults where available, informing the user of the values being used and their impact on the analysis.

To generate portions of the structural data required in the data deck, engineers often use independent software packages. For example, the nodes and elements of the finite element topology may be generated via a finite element preprocessor. To accommodate this practice, NESSUS/EXPERT must be able to read data in from existing files. Not all data will exist in files however, so NESSUS/EXPERT must also provide for manual entry of data. Regardless of the entry method used, NESSUS/EXPERT must allow the user to view and modify the data once it has been entered.

In serving as the front-end to the NESSUS structural analysis modules, NESSUS/EXPERT must provide guidance and advice to the user. This is probably the most important of NESSUS/EXPERT's functions given the size and complexity of this task. In order to best assist the user, NESSUS/EXPERT must

- prompt for all required information, being as specific as possible,
- activate and deactivate available options based on the current state of the job so that no incompatible selections are available to the user and so that all dependencies are represented,
- check for completeness of the data,
- check for inconsistencies between pieces of data,
- provide helpful hints involving idiosyncrasies of the NESSUS code and
- offer advice on optimal strategies in a given situation.

The final function required of NESSUS/EXPERT in its capacity as the front-end to the NESSUS structural analysis modules is to automatically generate the data deck needed by NESSUS from the data that has been entered. By providing the user with guidance and advice and preparing the data deck for the user, NESSUS/EXPERT will minimize the time spent by the engineer debugging the data deck.

2.1.2 An Intelligent Back-End Analysis Aid to NESSUS

The second major function of NESSUS/EXPERT is that it must serve as the back-end to the NESSUS structural analysis modules. The output from the structural analysis modules is simply a large file of numerical data. NESSUS/EXPERT must aid the user in analyzing and interpreting these results. Based on the results of the analysis, NESSUS/EXPERT must provide guidance to the user on what steps to take next. Recommendations might include modifying certain of the input data and rerunning the analysis to see the effects, or observing that a particular parameter is sensitive to certain variations in the probabilistic data. NESSUS/EXPERT must also provide helpful hints involving idiosyncrasies of the NESSUS code and optimal strategies in given situations.

2.2 The Challenge: Integrating AI Into The FORTRAN Engineering Environment

The types of knowledge that must be embodied in NESSUS/EXPERT given the functional requirements identified include both factual knowledge and knowledge which involves rules-of-thumb and heuristics related to experience gained in using the NESSUS system. Thus, the knowledge that had to be captured fit, in a fairly straightforward manner, the production rule knowledge representation technique. The challenge, however, came in satisfying the operational requirements involved with functioning in an engineering environment and still staying within time and budget constraints. These operational requirements are that NESSUS/EXPERT must run on a VAX under VMS, be able to run interactively, real-time on large amounts of data, must be packaged for easy distribution, and be delivered in FORTRAN-77 to insure portability and availability to the engineering community.

3 SELECTING AN APPROACH

3.1 Potential Approaches To The Development Of NESSUS/EXPERT

Given only the functional requirements of the system, the selected approach to the problem would have been simple enough since most expert system building tools support this type of knowledge representation scheme. However, since very few expert system building tools are capable of extensive interaction with external functions, especially those written in FORTRAN, it was necessary to identify and investigate the approaches that might satisfy both the functional and operational requirements.

One potential approach was to use an existing expert system building tool written in FORTRAN. At the start of this project over 2 years ago, there were only two such tools. These lacked sufficient sophistication in rule structures and manipulation due to the difficulties involved in performing inherently recursively-based tasks, such as parsing and analysis, in a non-recursive language. An added difficulty was that there were licensing problems involved in using a commercial product to develop a system that would be distributed within the government for free.

Another approach considered was to write our own expert system building tool in FORTRAN. Consideration was given to writing an OPS-like production system in FORTRAN, but it was determined that the effort to do an adequate job was well beyond the scope of the project. Such an effort would leave little time and money for the development of the expert system itself.

The possibility of writing the expert system itself in FORTRAN was also rejected. FORTRAN does not provide the non-algorithmic constructs and pattern matching capabilities needed for efficient expert system development. The effort required to develop an expert system in a language that does not provide these capabilities would be tremendous. Further arguments against using FORTRAN as the development language for NESSUS/EXPERT were that it would probably be unacceptably inefficient and would lose many of the primary benefits of AI technology, such as flexibility, expandability, and maintainability.

The fourth option considered was to use an existing non-FORTRAN-based expert system building tool that could interface to FORTRAN and address the problem of delivery in FORTRAN later in the project. This temporary solution would allow for progress to be made in the development of the expert system thus enabling a better understanding of the effects the operational require-

ments were going to have on the system. A search was made to find an existing, inexpensive, fairly well-supported expert system building tool that could aid in the writing of production rules as well as readily interface to FORTRAN code.

No such system was found. Thus, the requirement that the tool be able to interface with FORTRAN code was reduced to being able to interface with the operating system and/or file system. Thus a public domain version of OPS5 was chosen for experimentation. This version was written in Franz LISP so the environment was not ideal. However, it did run on a VAX and it allowed rapid prototyping thus facilitating time effective experimentation with the interface between the expert system and the existing NESSUS code. The interface with FORTRAN was simulated using access to files. OPS5 proved to be sufficient in terms of the functional needs of the expert system itself but, as expected, not in the operational issues of integration and delivery.

Shortly after the development of the NESSUS/EXPERT prototype, a version of OPS5 written in BLISS was offered by DEC. This version provided the flexibility and power of the OPS5 language but not the problems involved with running in the Lisp environment. DEC OPS could interface reasonably well with the DEC FORTRAN on the VAX. The prototype of NESSUS/EXPERT was converted to DEC OPS to test the FORTRAN communication capabilities.

The resulting system was satisfactory from the standpoint of integration into the FORTRAN environment. However, inefficiency of the DEC OPS/FORTRAN interface was a concern as were the licensing issues involved with using a commercial tool to develop an expert system that was to be embedded in an existing FORTRAN program and distributed to engineering environments where DEC OPS would most likely not be available.

During this time vendors began offering a few tools that could access non-Lisp environments. This capability was made possible because the tools were not written in Lisp, but in more conventional programming languages like C. These included S.1, M.1, various versions of OPS, and CLIPS[1]. Of these CLIPS(C Language Integrated Production System) provided the most benefits for the project. The benefits of CLIPS were that it was not a commercial tool, it was developed at Johnson Space Center and was readily available to other NASA projects, it was written in C and thus highly portable, it could easily integrate into the FORTRAN environment, and access was available to both the source code and the people who wrote it.

3.2 The Selected Approach - CLIPS and FORTRAN

Some preliminary implementations, tests, and experiments were run to test the integration of CLIPS with FORTRAN and to determine the amount of effort required to convert the now quite large prototype of NESSUS/EXPERT from OPS to CLIPS. The results were very positive. Therefore, if NESSUS/EXPERT were implemented in CLIPS the sole remaining deficiency would be the requirement for delivery in FORTRAN. Since the primary reason for requiring delivery in FORTRAN had been to insure portability and availability to NASA, the NASA sponsor of the PSAM project consented to delivery in CLIPS. Therefore, CLIPS was selected as the implementation language for NESSUS/EXPERT.

In light of the FORTRAN interfacing capabilities that CLIPS provides, a reassessment of the NESSUS/EXPERT design was made. In previous implementations all the required NESSUS/EXPERT functions had to be implemented with the rule-based tool, including those tasks that were best suited for conventional programming methods. With CLIPS' ability to integrate with FORTRAN, an investigation was made into whether the non-AI type tasks should be moved from the rules out to FORTRAN routines.

The NESSUS/EXPERT functions were divided into two classes: functions that performed high-level tasks that are well-suited to AI solutions, such as decision making, consistency checking and evaluation and functions that performed lower-level functions which could run more efficiently implemented in FORTRAN.

The NESSUS/EXPERT tasks identified as best suited for CLIPS implementation include consistency and completeness checking, inference of data from other available data, interpretation and analysis of data, providing guidance and advice on the use of the system, and control tasks, such as running the dynamic menu system for the user interface. These tasks require either heuristic-type knowledge or knowledge concerning the relationships between the categories of NESSUS data and are well suited to implementation as forward chaining rule bases.

The NESSUS/EXPERT tasks identified as best suited for FORTRAN implementation include user I/O, file I/O, editor-type tasks, such as type and range checking, mathematical computations, and data deck generation. It is worthy of note that although the functions selected for FORTRAN implementation can execute more efficiently written in FORTRAN, the development of FORTRAN code for these functions required considerably more time than it had with the expert system building tool. These FORTRAN routines are also much less flexible than the corresponding OPS5 rules which performed basically the same functions.

The resulting system brings the best of both conventional and AI programming techniques together. CLIPS rules drive the NESSUS/EXPERT system. The CLIPS rules invoke the appropriate FORTRAN modules to perform low-level functions as needed. The CLIPS rules pass data to C interface routines which pass the data on to the FORTRAN modules as calling parameters. The FORTRAN modules make assertions into the CLIPS world via a handful of FORTRAN and C interface routines. The interactions between the CLIPS rules and the FORTRAN modules are represented in Figure 1.

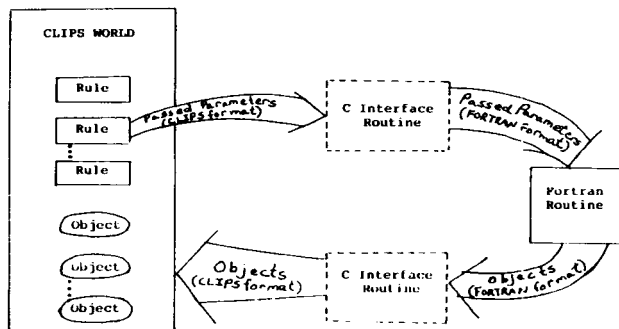


FIGURE 1. CLIPS/FORTRAN COMMUNICATION

NESSUS/EXPERT communicates with the user through both the CLIPS rules and the FORTRAN modules. The design of the interface is such that it is not apparent to the user whether CLIPS rules or FORTRAN code is driving it at any given time. The FORTRAN modules provide the interface to the NESSUS modules and the user's data files. Figure 2 shows the external interfaces to NESSUS/EXPERT.

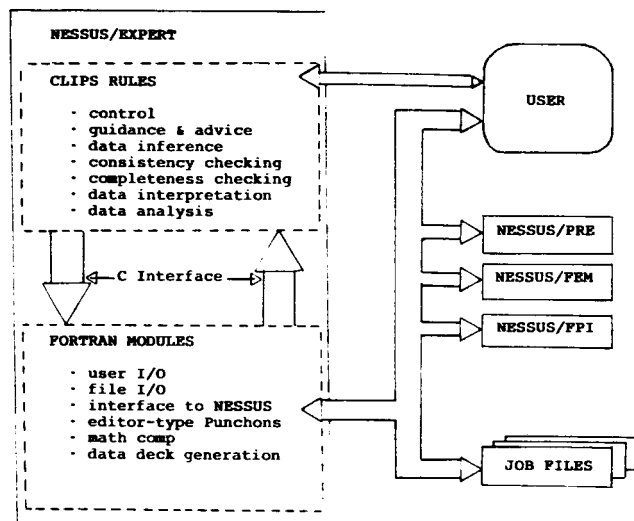


FIGURE 2. NESSUS/EXPERT EXTERNAL COMMUNICATION

4 OVERVIEW OF NESSUS/EXPERT

Currently the system consists of around 700 CLIPS rules and 300 FORTRAN routines. It runs on a VAX under VMS. A partial example session for building a deterministic data deck is presented in the appendix.

The marriage of CLIPS and FORTRAN provided several capabilities in terms of satisfying the operational requirements of NESSUS/EXPERT. The efficiency of the system is optimized because each of the languages performs the types of tasks which they do best. Well organized rule sets and FORTRAN modules provide modularity. The use of CLIPS and FORTRAN77 provides for both portability and availability to the engineering community. Thus the operational requirements of the system were satisfied.

Many of the functional requirements are satisfied via a "smart", CLIPS controlled menu system which is the basis of the user interface. Knowledge of the process that the engineer must follow to use NESSUS is encoded into the menu system. Figure 3 shows the hierarchical structure of the menus in the NESSUS/EXPERT system.

In a typical session, NESSUS/EXPERT would first present the 'JOB SELECTION' menu. The job selected specifies which structural analysis problem the user would like to work on. The user may select to continue work on a job that has been previously worked on or enter a new jobname to begin work on a new structural analysis problem. When the job is specified, NESSUS/EXPERT automatically retrieves all of the information that has previously been entered for that job. Once all of the information for the selected job has been retrieved, NESSUS/EXPERT presents the user with the 'ACTIVITY SELECTION' menu. Like most of the NESSUS/EXPERT menus, the contents of this menu are dynamically determined by NESSUS/EXPERT based on the current status of the job. For example, the activity selection menu for a job will include the option to run a deterministic analysis only if a complete and consistent deterministic datadeck exists for the job. From the 'ACTIVITY SELECTION' menu the user can move throughout the menu system as desired. The only restriction on movement through the menu system is that NESSUS/EXPERT will not provide access to menus which do not make sense in the current context.

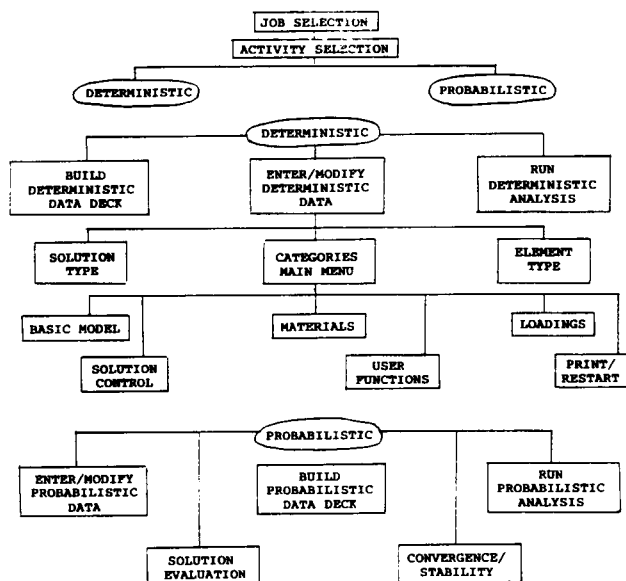


FIGURE 3. NESSUS/EXPERT MENU STRUCTURE

Within the NESSUS/EXPERT menu system, there are five basic types of screens. These five types of screens are used for five different types of interactions with the user. Consistent formats have been defined for each of the five screen types to enhance user-friendliness. These five screen types include a menu screen, a manual entry screen, a flag entry screen, a help screen, and an information screen.

The menu screen type is used by NESSUS/EXPERT to prompt the user for a choice between several available activity options. The CLIPS rules control when the menu screen appears and what items appear on it. The FORTRAN modules operate the actual input and output for the screen. Figure 4 exemplifies the NESSUS/EXPERT menu screen. The following features are common to all menu screens.

- A title on the top line indicates both the current orientation within the NESSUS/EXPERT system and the purpose of the screen.
- The name of the current job appears on the top line next to the title.
- A brief set of instructions appears after the title line.
- A HELP option is provided as a menu item. When this option is selected, NESSUS/EXPERT displays one or more screens of text which explain the current menu. When the user exits the HELP screen, NESSUS/EXPERT returns to the menu from which the HELP option was selected. This provides an on-line help facility which is context sensitive. The types of knowledge available via this help facility include knowledge about the use of NESSUS/EXPERT and knowledge about NESSUS. It serves as an enhanced, automated NESSUS user's manual.
- A QUIT option is provided as a menu item. Selection of this option usually returns the user to the next level up in the menu hierarchy.

Manual entry screens provide an editor-like means for the user to enter data directly into the NESSUS/EXPERT system with insurance that the data meets the requirements of the NESSUS modules. The manual entry screens allow the user to enter, delete, modify and view data. Manual entry screens are run by FORTRAN modules. They are invoked by CLIPS rules at the appropriate times. The FORTRAN manual entry modules contain specific

*** BASIC MODEL MENU FOR EXAMPLE ***

SELECT THE CATEGORY OF DATA TO BE ENTERED.
(Categories marked with * have been at least partially entered.)

1 QUIT	2 *COORDINATES	3 *ELEMENTS
4 BOUNDARY	5 TRANSFORMATIONS	6 TYING
7 SPRINGS	8 MASSES	9 DASHPOTS
10 DUPLICATENODE	11 EMBED	12 HELP

>> 2

FIGURE 4. TYPICAL NESSUS/EXPERT MENU SCREEN

knowledge about types and ranges required for each piece of data within each category. Where these types and ranges vary depending on other data values that have been entered, the invoking CLIPS rules inform the FORTRAN modules of the appropriate restrictions. Figure 5 exemplifies the NESSUS/EXPERT manual entry screen. The following features are common to all manual entry screens.

- A title on the top line indicates the category and the job name.
- The most recently manipulated lines of data are displayed.
- A brief set of instructions follow the data display.
- The user can return to the previous menu by typing 'q'.
- The help facility (as described above) can be accessed by typing 'h'.
- A line of data can be deleted by typing 'd' and the line number.
- A line of data can be viewed by typing the line number.
- A line of data can be changed by simply reentering the line.

*** COORDINATES MANUAL ENTRY FOR EXAMPLE ***

Coordinates data for line 1 and subsequent lines:

1)	1	0.0000	0.0000	0.0000	0.1000
2)	2	1.0000	0.0000	0.0000	0.1000
3)	3	2.0000	0.0000	0.0000	0.1000
4)	4	3.0000	0.0000	0.0000	0.1000
5)	6	5.0000	0.0000	0.0000	0.1000
6)	7	6.0000	0.0000	0.0000	0.1000
7)	8	7.0000	0.0000	0.0000	0.1000
8)	9	8.0000	0.0000	0.0000	0.1000

Enter an existing node number and 4 new data values,
new node number and 4 data values or a
line number (to view existing coordinates data).

(Enter Q(uit) when all COORDINATES data has been entered.)

-3 1 2 3 4

FIGURE 5. TYPICAL NESSUS/EXPERT MANUAL ENTRY SCREEN

Flag screens provide a method of manual entry for information which NESSUS uses that does not involve data values. The manual entry screen allows pieces of information to be toggled on and off. The CLIPS rules control when the flag screens appear and the FORTRAN modules perform the actual input and output for the screen. Figure 6 exemplifies the NESSUS/EXPERT flag screen. The following features are common to all flag screens.

- A title on the top line indicates the category and the job name.
- A brief set of instructions follows the title line
- The first option (quit) allows the user to leave the flag screen.
- The second option (on) allows for the flag to be activated.
- The third option (off) allows for the flag to be deactivated.
- The fourth option (help) provides access to the help facility as described above.

ORIGINAL PAGE IS OF POOR QUALITY

*** SET/RESET FLAG FRONTALSOLUTION ***

Flag is currently SET OFF.

- 1 Quit
 - 2 SET flag to ON
 - 3 RESET flag to OFF
 - 4 HELP
- >> 2

FIGURE 6 - TYPICAL NESSUS/EXPERT FLAG SCREEN

The help screen is used by NESSUS/EXPERT to display text explanations to the user. The contents of the help screen will apply specifically to the user's current position within the menu system and to the current status of the job. The types of knowledge available via this help facility include knowledge about the use of NESSUS/EXPERT and knowledge about NESSUS. It serves as an enhanced, automated NESSUS user's manual. Help screens explain both what is expected of the user under the current circumstances and additional information concerning the technical aspects of the subject at hand. This provides an on-line help facility which is context sensitive. Help screens are provided by the FORTRAN modules. When the user exits the HELP screen, NESSUS/EXPERT returns to the screen from which the HELP option was selected. Figure 7 exemplifies the NESSUS/EXPERT help screen. The following features are common to all help screens.

- A title on the top line indicates the topic of help and the job name.
- Striking the return key will either display more help if available or will return the user to the previous screen.

*** ELEMENT TYPE HELP SCREEN ***

Select the element type to be used in the analysis by entering the number which corresponds to that element. The element types currently available are:

- 3: 4-node plane-stress. Like MARC element type 3.
- 7: 8-node solid. Like MARC element type 7.
- 10: 4-node axisymmetric solid. Like MARC element type 10.
- 11: 4-node plane strain. Like MARC element type 11.
- 75: 4-node thick/thin shell. Like MARC element type 75.
- 98: 2-node Timoshenko beam. Like MARC element type 98.
- 151: 4-node assumed strain plane stress.
- 152: 4-node assumed strain plane strain.
- 153: 4-node assumed strain axisymmetric.
- 154: 8-node assumed strain solid.

The current version of MHOST only allows finite element models consisting of a single element type.

(RETURN to continue)

FIGURE 7. TYPICAL NESSUS/EXPERT HELP SCREEN

The information screens provide high-level information to the user, such as warnings and advice. For example, NESSUS/EXPERT would use an information screen to alert the user when an inconsistency is detected in the model data. Figure 8 exemplifies the NESSUS/EXPERT information screen. The following features are common to all information screens.

- A title on the top line indicates the type of information involved and the job name.
- The information itself is displayed as the body of the screen.
- A list of the options available to the user in response to the given information is presented after the information.

Thus, NESSUS/EXPERT appears to the user as a set of highly flexible menu interfaces. In general, the appearance of a menu and the options available on it are determined by rules written in CLIPS that utilize knowledge about the current state of the job and what the user wishes to do. CLIPS passes any directions concerning what should be displayed to the appropriate FORTRAN routines. The

*** CONSISTENCY CHECK FOR EXAMPLE ***

Element data references the following undefined nodes.

5

TYPE C to enter coordinate data for the node(s) OR
E to change the element definition(s).
I to ignore the inconsistency for now.

>> i

FIGURE 8 - TYPICAL NESSUS/EXPERT INFORMATION SCREEN

FORTTRAN routines handle the actual display of the menu and the acceptance of the input from the user. The FORTRAN routines then return any information that is needed for higher level decisions back to the CLIPS environment.

5 CONCLUSIONS

Over the past few years artificial intelligence, especially expert system technology, has been applied to a wide variety of real world problems. At the same time, it has become apparent that in order for such systems to be truly useful, they must be able to integrate with other real world computer software systems. The level of integration required by any given system depends on its needs. The need to access certain data, either from a flat file or database, is fairly simple and straightforward. The need to share control and functionality, on the other hand, is not so easy.

NESSUS/EXPERT must not only access data, it must access and analyze a large amount of data in real time. The analysis that must be performed usually depends not on the data itself, but on certain attributes of the data, such as the number of parameters or the maximum value of a certain parameter. Processing the data and calculating these attributes is more efficiently handled by a conventional language such as FORTRAN. Analyzing these attributes and developing conclusions is better handled by a higher level language such as CLIPS.

Aside from the issue of data access, NESSUS/EXPERT must also be capable of invoking the NESSUS code itself, thus serving as a user interface to the structural analysis system. Also, due to the need to divide the tasks between FORTRAN and CLIPS based on functionality, NESSUS/EXPERT must be able to invoke both FORTRAN and CLIPS routines in as seamless a manner as possible. Therefore, NESSUS/EXPERT must manage control and functionality between the FORTRAN and CLIPS environments.

We have been successful in achieving the requirements for integration between an artificial intelligence and an engineering environment in NESSUS/EXPERT for several reasons. First, an appropriate division of functional tasks was defined. These tasks were then associated with the proper method for implementation, artificial intelligence or conventional. Based on the amount and type of integration required as a result of assigning tasks to implementation approaches and the other requirements of the project, an appropriate set of tools was selected, in this case CLIPS and FORTRAN. Finally, the actual integration routines were designed and implemented to insure as seamless a transition between the two environments as possible. In this way a system that takes advantage of the best in both the artificial intelligence and engineering environments can be successfully developed in an efficient and effective manner.

6 REFERENCES

- [1] Culbert, Chris, "CLIPS Reference Manual - Version 4.0," Mission Support Directorate, MPAD, NASA-JSC, March 1987.

ORIGINAL PAGE IS OF POOR QUALITY

[2] Nakazawa, Dias, Nagtegaal, and Wertheimer, "MHOST Users' Manual - Version 3.3," April 1986.

APPENDIX - A SAMPLE SESSION

The first screen displayed by the NESSUS/EXPERT system is the JOB SELECTION menu (Screen 1). This menu allows the user to select from the jobs that have been previously worked on via the NESSUS/EXPERT system or to enter the name of a new job. Notice in Screen 1 a new job called EXAMPLE has been selected rather than selecting an already existing job.

Next NESSUS/EXPERT prompts the user for a comment that will be placed at the top of the data file for future reference. Entry of the comment is optional and is provided only for the user's convenience. Screen 2 shows the COMMENT ENTRY screen.

In most cases, NESSUS/EXPERT allows the user to determine the order in which various types of data are entered. However, there are two pieces of data that NESSUS/EXPERT requires to be entered before all others. These are the analysis type and the element type. NESSUS/EXPERT requires the entry of these before all other data due to the high degree of dependencies of the subsequent data requirements imposed by these two pieces of data. In Screen 3 STATIC is chosen for the analysis type.

In Screen 4, the element type is prompted for and the HELP option has been selected. This results in the display of a help screen for selection of the NESSUS element type (Screen 4).

In this sample session, the return key was hit to return to the element type selection menu and element type "75" was selected. Thus with the mandatory pieces of data having been entered, the SYSTEM MENU for the job is displayed (Screen 6). The options in this menu are generated dynamically based on the current state of the job. Notice there are only three options available to the user at this point. The user must enter the minimum data requirements for the deterministic datadeck before other options will be offered. This is simply because no other options make sense until the deterministic data has been entered. For this sample session, the option to enter the deterministic model data is selected.

There are around 70 categories of deterministic data that can be entered via NESSUS/EXPERT. Therefore, these categories have been organized into 6 separate menus for ease of use. The menu shown in Screen 7 prompts the user for selection of one of these 6 menus. For our sample session the BASIC MODEL MENU has been selected.

Screen 8 shows the categories offered by the BASIC MODEL MENU. In this screen, the user has selected to enter the deterministic data for the COORDINATES category.

Most category data can be entered either from an external file or by manual entry. (If the category was previously entered, it does not have to be reentered. The system will automatically read in all data

```
*** JOB SELECTION ***
SELECT THE JOB YOU WOULD LIKE TO WORK ON.
(TO begin work on a new job, enter the new jobname.)
(TO delete a job, type "D <return>" or "d <return>".)

1 QUIT          2 YKP          3 SAMPLE
4 VALID CASE3   5 HELP

>> EXAMPLE
```

SCREEN 1

```
*** COMMENT ENTRY FOR EXAMPLE ***
ENTER THE COMMENT FOR THE JOB.
(The comment must be strictly alphanumeric;
no special characters.)

Current comment:
```

>> Example job for demonstration purposes

SCREEN 2

```
*** ANALYSIS TYPE SELECTION FOR EXAMPLE ***
SELECT THE ANALYSIS TYPE YOU WOULD LIKE MODEL.
```

```
1 static          2 dynamic
3 frequency       4 buckling
5 modal          6 frequency domain dynamic
7 HELP
```

>> 1

SCREEN 3

```
*** ELEMENT TYPE MANUAL ENTRY FOR EXAMPLE ***
Enter the element type.
```

```
1 QUIT    2 "3"    3 "7"    4 "10"   5 "11"
6 "75"    7 "98"   8 "151"  9 "152" 100 "153"
11 "154" 12 HELP
```

>> 12

SCREEN 4

```
*** ELEMENT TYPE HELP SCREEN ***
```

Select the element type to be used in the analysis by entering the number which corresponds to that element. The element types currently available are:

```
3: 4-node plane-stress. Like MARC element type 3.
7: 8-node solid. Like MARC element type 7.
10: 4-node axisymmetric solid. Like MARC element type 10.
11: 4-node plane strain. Like MARC element type 11.
75: 4-node thick/thin shell. Like MARC element type 75.
98: 2-node Timoshenko beam. Like MARC element type 98.
151: 4-node assumed strain plane stress.
152: 4-node assumed strain plane strain.
153: 4-node assumed strain axisymmetric.
154: 8-node assumed strain solid.
```

The current version of MHOST only allows finite element models consisting of a single element type.

(RETURN to continue)

SCREEN 5

```
*** SYSTEM MENU FOR EXAMPLE ***
```

SELECT AN ACTIVITY.

```
1 Exit the PSAM Expert system.
2 Return to JOB SELECTION MENU.
3 Enter/Alter/Complete deterministic model data.
4 HELP
```

>> 3

SCREEN 6

```
*** MODEL DATA GROUPS MENU FOR EXAMPLE ***
SELECT THE DESIRED MODEL DATA MENU.
```

```
1 BASIC MODEL MENU    2 MATERIAL DATA MENU
3 LOADINGS MENU       4 USER FILE SPECS MENU
5 SOLUTION CONTROL MENU 6 PRINTOUT CONTROL MENU
7 Return to SYSTEM MENU 8 HELP
```

>> 1

SCREEN 7

associated with a job when the job is selected.) In Screen 9, the user has specified that the data be read in from a file called COORD.DAT.

When the coordinates data has been read in the system returns to the BASIC MODEL MENU for the next selection. Notice in Screen 10 that the COORDINATES option has been marked with an asterisk to indicate that this data has been entered. The elements data has also been entered in the same way although for the sake of brevity this process was not shown here. In Screen 10, the user is reselecting the COORDINATES option.

Since some COORDINATES data has already been entered, the system takes the user directly to manual entry. In Screen 11, the user has entered a bad data value (an illegal node number) for new coordinates data.

Screen 12 shows an error message resulting from the entry in Screen 10. The user now has entered "d 5" to tell the system to delete the 5th line of data.

Screen 13 shows that the data from line 5 was deleted and the remaining data was shifted to fill that line. The user has now selected to QUIT from COORDINATES manual entry.

Once the COORDINATES manual entry is left, the CLIPS rules detect that there is an unusual situation. Screen 14 shows an information screen which alerts the user to the fact that the nodes defined are not in the expected consecutive sequence. In this case, the user assures the system that this was the intention and the system forgets the matter. Had the user answered NO to the query, the system would have returned the user to manual entry for correction of the problem.

Screen 15 shows a second information screen presented by the CLIPS rules. Here the system has detected an unacceptable inconsistency between the ELEMENTS data and the COORDINATES data. Selection of either the C or E option would take the user directly to manual entry for the data type. In the example the user has selected to ignore the inconsistency for the time being. The NESSUS/EXPERT system will not forget that this inconsistency exists, however. The user will be reminded of it before a deterministic data deck is built.

Again, for the sake of brevity, screens are not provided for the entry of the remaining required deterministic data. In Screen 16, the user has moved to the SOLUTION CONTROL MENU. The FRONTAL-SOLUTION option has been chosen. Notice that there are 13 categories of data in this menu and none have been previously selected (no asterisks).

The FRONTALSOLUTION category is simply a flag to NESSUS, therefore no data is required. The category is either present or not

```

*** BASIC MODEL MENU FOR EXAMPLE ***
SELECT THE CATEGORY OF DATA TO BE ENTERED.
(Categories marked with * have been at least
partially entered.)

1  QUIT                2  COORDINATES          3  ELEMENTS
4  BOUNDARY           5  TRANSFORMATIONS      6  TYING
7  SPRINGS            8  MASSES                9  DASHPOTS
10 DUPPLICATENODE    11  EMBED                  12  HELP

>> 2
SCREEN 8

```

```

*** COORDINATES ENTRY METHOD MENU FOR EXAMPLE ***
SELECT A DATA ENTRY METHOD.

1  QUIT
2  Enter data by hand.
3  Read data in from file. (Enter filename)
4  HELP

>> coord.dat

```

```

SCREEN 9
*** BASIC MODEL MENU FOR EXAMPLE ***
SELECT THE CATEGORY OF DATA TO BE ENTERED.
(Categories marked with * have been at least
partially entered.)

1  QUIT                2  *COORDINATES          3  *ELEMENTS
4  BOUNDARY           5  TRANSFORMATIONS      6  TYING
7  SPRINGS            8  MASSES                9  DASHPOTS
10 DUPPLICATENODE    11  EMBED                  12  HELP

>> 2

```

```

SCREEN 10
*** COORDINATES MANUAL ENTRY FOR EXAMPLE ***
Coordinates data for line 1 and subsequent lines:

1) 1 0.0000 0.0000 0.0000 0.1000
2) 2 1.0000 0.0000 0.0000 0.1000
3) 3 2.0000 0.0000 0.0000 0.1000
4) 4 3.0000 0.0000 0.0000 0.1000
5) 5 5.0000 0.0000 0.0000 0.1000
6) 7 6.0000 0.0000 0.0000 0.1000
7) 8 7.0000 0.0000 0.0000 0.1000
8) 9 8.0000 0.0000 0.0000 0.1000

Enter an existing node number and 4 new data values,
new node number and 4 data values or a
line number (to view existing coordinates data).

(Enter Q(uit) when all COORDINATES data has been entered.)
-3 1 2 3 4

```

```

SCREEN 11
*** COORDINATES MANUAL ENTRY FOR EXAMPLE ***
INVALID NODE NUMBER ENTERED.
LEGAL RANGE: 1 TO 400.

Coordinates data for line 1 and subsequent lines:

1) 1 0.0000 0.0000 0.0000 0.1000
2) 2 1.0000 0.0000 0.0000 0.1000
3) 3 2.0000 0.0000 0.0000 0.1000
4) 4 3.0000 0.0000 0.0000 0.1000
5) 6 5.0000 0.0000 0.0000 0.1000
6) 7 6.0000 0.0000 0.0000 0.1000
7) 8 7.0000 0.0000 0.0000 0.1000
8) 9 8.0000 0.0000 0.0000 0.1000

Enter an existing node number and 4 new data values,
new node number and 4 data values or a
line number (to view existing coordinates data).

(Enter Q(uit) when all COORDINATES data has been entered.)
d 5

```

```

SCREEN 12
*** COORDINATES MANUAL ENTRY FOR EXAMPLE ***
Coordinates data for line 5 and subsequent lines:

5) 6 5.0000 0.0000 0.0000 0.1000
6) 7 6.0000 0.0000 0.0000 0.1000
7) 8 7.0000 0.0000 0.0000 0.1000
8) 9 8.0000 0.0000 0.0000 0.1000
9) 10 9.0000 0.0000 0.0000 0.1000
10) 11 10.0000 0.0000 0.0000 0.1000
11) 12 0.0000 1.0000 0.0000 0.1000
12) 13 1.0000 1.0000 0.0000 0.1000

Enter an existing node number and 4 new data values,
new node number and 4 data values or a
line number (to view existing coordinates data).

(Enter Q(uit) when all COORDINATES data has been entered.)
q

```

```

SCREEN 13

```

present. Screens 17 and 18 demonstrate how NESSUS/EXPERT provides for this type of category to be turned on and off. In the example, the category is turned on.

Screen 19 shows the SOLUTION CONTROL MENU consequent to the FRONTSOLUTION option being selected. Notice that there are now only 12 categories of data in this menu. NESSUS/EXPERT has removed the BANDMATRIX option from the menu. This is because FRONTSOLUTION and BANDMATRIX are mutually exclusive options for the NESSUS input.

Screen 20, the final screen in this example shows that once the minimum data requirements for the deterministic model have been entered, options are added to the system menu accordingly.

```
*** COORDINATES MANUAL ENTRY FOR EXAMPLE ***
Undefined COORDINATES NODES:
5
Should these NODES be left undefined (Enter Y or N).
Y
SCREEN 14
```

```
*** CONSISTENCY CHECK FOR EXAMPLE ***
Element data references the following undefined nodes.
5
TYPE C to enter coordinate data for the node(s) OR
      E to change the element definition(s).
      I to ignore the inconsistency for now.
>> i
SCREEN 15
```

```
*** SOLUTION CONTROL MENU FOR EXAMPLE ***
SELECT THE CATEGORY OF DATA TO BE ENTERED.
(Categories marked with * have been at least
partially entered.)

1  QUIT                2  ITERATIONS
3  INCREMENTS          4  TIME
5  BANDMATRIX          6  BFGS
7  DEFORMATION         8  DISPLACEMENTMETHOD
9  FRONTSOLUTION       10 LINESEARCH
11 LOUBIGNAC           12 OPTIMIZE
13 SECANTNEWTON        14 TANGENT
15 HELP

>> 9
SCREEN 16
```

```
*** SET/RESET FLAG FRONTSOLUTION
Flag is currently SET OFF.

1  Quit
2  SET flag to ON
3  RESET flag to OFF
4  HELP

>> 2
SCREEN 17
```

```
*** SET/RESET FLAG FRONTSOLUTION
Flag is currently SET ON.

1  Quit
2  SET flag to ON
3  RESET flag to OFF
4  HELP

>> 1
SCREEN 18
```

```
*** SOLUTION CONTROL MENU FOR EXAMPLE ***
SELECT THE CATEGORY OF DATA TO BE ENTERED.
(Categories marked with * have been at least
partially entered.)

1  QUIT                2  ITERATIONS
3  INCREMENTS          4  TIME
5  BFGS                6  DEFORMATION
7  DISPLACEMENTMETHOD 8  *FRONTSOLUTION
9  LINESEARCH          10 LOUBIGNAC
11 OPTIMIZE            12 SECANTNEWTON
13 TANGENT             14 HELP

>> 1
SCREEN 19
```

```
*** SYSTEM MENU FOR EXAMPLE ***
SELECT AN ACTIVITY.

1  Exit the PSAM Expert system.
2  Return to JOB SELECTION MENU.
3  Change job comment.
4  Reselect analysis type.
5  Enter/Alter/Complete deterministic model data.
6  Enter/Alter/Complete random variables.
7  Build Deterministic Datadeck.
8  Run deterministic analysis.
9  HELP

>> 6
SCREEN 20
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPLYING AI TO REAL WORLD ENGINEERING PROBLEMS

Carol J. Russo
General Electric

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

A LOW COST PORTABLE TELE-AUTONOMOUS MAINTENANCE STATION

Michael W. Walker

Shih-Ying Sheu

Richard A. Volz

Lynn Conway

Robotics Research Laboratory

Department of Electrical Engineering and Computer Science

The University of Michigan

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

KNOWLEDGE-BASED ZONAL GRID GENERATION FOR COMPUTATIONAL FLUID DYNAMICS

Alison E. Andrews
Applied Computational Fluids Branch
Mail Stop 258-1
NASA Ames Research Center
Moffett Field, California 94035

ABSTRACT

Automation of flow field zoning in two dimensions is an important step towards reducing the difficulty of three-dimensional grid generation in computational fluid dynamics. Using a knowledge-based approach makes sense, but problems arise which are caused by aspects of zoning involving perception, lack of expert consensus, and design processes. These obstacles are overcome by means of a simple shape and configuration language, a tunable zoning archetype, and a method of assembling plans from selected, predefined subplans. A demonstration system for knowledge-based two-dimensional flow field zoning has been successfully implemented and tested on representative aerodynamic configurations. The results show that this approach can produce flow field zonings which are acceptable to experts with differing evaluation criteria.

INTRODUCTION

Computational fluid dynamics (CFD) is becoming an essential tool in the understanding of fluid physics and in the design of aerospace vehicles. The long range goal is to be able to quickly and accurately simulate the viscous flow about realistic configurations¹. Ideally, the time required for each computation should be small enough to permit incorporation of such a solution technique into the design cycle. Several aspects of CFD have been identified as pacing items, or areas which require significant advances before the goals of CFD can be realized^{2,3}. Three-dimensional grid generation is prominent among them. Although grid generation methods have become fairly sophisticated, it is often extremely difficult to generate a reasonable single grid about a general, three-dimensional configuration⁴⁻⁶. Factors which are principally responsible for this difficulty are: 1) complex geometries, 2) the need for selective refinement to resolve fluid physics phenomena efficiently, and 3) limitations on the size of computer physical memory. Domain decomposition, or flow field zoning, is an effective solution. Partitioning a flow field into zones can reduce the topological complexity of the problem. It allows grid refinement to be limited to just those regions of the flow field where it is required in order to resolve high gradients. Also, if each zonal grid is small enough to fit into the physical memory of the computer, problems of any size can be tackled.

The full benefits of flow field zoning can be gained only if it is done well. A user must have the following qualifications: experience with zonal methods, familiarity with grid generation capabilities, fluid dynamics knowledge, knowl-

edge about the flow solver being used, criteria for evaluating zonings, and imagination. Unfortunately, the following conditions exist which prevent the widespread use of a zonal approach: 1) those qualifications are possessed by few, and are not easily taught; 2) tedium and frequent error are inherent in zonal boundary condition specification, regardless of the user's expertise; 3) it is difficult even for an expert to visualize and specify general zonal boundary surfaces in three dimensions; and 4) zoning evaluation criteria are not well established. Clearly, automating flow field zoning is an important step towards achieving routine application of CFD to aerospace problems⁴.

The goal of the present research is to lay the foundation for an automated zonal grid (composite grid) generation capability in three dimensions by developing a knowledge-based demonstration program capable of automated flow field zoning in two dimensions. There are aspects of the zoning problem which make a knowledge-based programming approach a natural choice. There are other aspects, however, which tend to discourage the use of knowledge-based techniques as they currently exist. The purpose of this paper is to briefly describe the demonstration system which has been implemented, focusing on the methods used to overcome the difficulties arising from these latter aspects.

EZGRID

A knowledge-based system called EZGrid (Expert Zonal Grid generator) has been developed to demonstrate the feasibility of using a knowledge-based approach to automate flow field zoning for typical two-dimensional CFD applications. The program was written in MRS^{7,8} (a logic programming language developed at Stanford University), Franz Lisp, and C. It contains over 400 rules which enable it to automatically design flow field zonings and generate the necessary zonal boundary curves for representative two-dimensional aerodynamic configurations. The input consists of one or more geometry coordinate data files and the desired inflow conditions (Mach number, angle of attack, etc.). Zonal boundary coordinate data files are output in a form which can be accepted directly by a grid generator. Approximately three person-years were required for the development of EZGrid.

Figure 1 shows a simple zonal grid (i.e., that has only one zone) for a single airfoil in a subsonic, inviscid fluid at 4° angle of attack. In simple cases such as this, EZGrid can automatically set the grid generation parameters as well, so this grid was generated without human intervention. In general, EZGrid provides the boundaries and

topologies for each zonal grid, and the user sets the grid generation parameters for the final step of grid generation. Figure 2 shows the seven-zone (or seven-block) grid designed by EZGrid for a four-element airfoil in a transonic, viscous fluid at 4° angle of attack. All of the grids in this paper were generated using the General Dynamics grid generator GRIDGEN2D⁹. Note that the airfoil in Figure 1 is identical to the downstream-most airfoil in Figure 2, demonstrating that in addition to shape and inflow conditions, the configuration plays an important role in determining the zoning.

EZGrid has both automatic and interactive modes of operation. Though an automated design is possible for a variety of zoning problems, a user may choose to design a zoning interactively (to test out a particular idea, for example).

A KNOWLEDGE-BASED APPROACH

There are several useful rules for identifying tractable task domains for knowledge-based system development¹⁰, which are: 1) a closed-form or algorithmic solution cannot be found, 2) domain expertise exists, 3) the task can be performed by an expert within a reasonable amount of time (hours, days), 4) the task is cognitive (as opposed to perceptual), 5) the skill is routinely taught to nonexperts, and 6) the task is worth doing, and has a high payoff. Furthermore, a distinction is often made between problems which are solved by means of classification, or selection of predefined solutions, and problems whose solutions must be constructed or designed (analysis versus synthesis)¹¹. Problems that require construction of a solution usually respond to knowledge-based techniques less readily than those which can be solved using selection or classification procedures.

Several aspects of flow field zoning obey these rules. Zoning is an ill-structured problem, and no satisfactory conventional solution has been found. Expertise is required to perform the task well, and an expert can design and generate a zoning in several days or weeks, depending on the complexity of the configuration. Finally, flow field zoning is an important element in the drive to make three-dimensional grid generation faster and easier, and is thus definitely worth doing.

Unfortunately, several aspects of flow field zoning break these rules. The art of flow field zoning is not easily taught, perhaps because there is no good language to describe the process. The task has an unmistakable perceptual element, involving qualitative shape and position information. The process of flow field zoning has been modelled as one in which a solution is designed as opposed to selected. Lastly, while there are recognized experts, ideas differ (and are even still evolving) as to what constitutes a good zoning, so the solution preferred by one expert may be unacceptable to another. This implicit "user bias" affects the design and evaluation of flow field zonings. These aspects of flow field zoning serve to make the application of knowledge-based techniques challenging at best.

The plan that was adopted to successfully develop a system for this domain included the following:

1. Develop a model and language to describe the fundamentals of two-dimensional flow field zoning.
2. Debug the model and language through the implementation of an *interactive* knowledge-based system (in which the mechanics and bookkeeping of designing a zoning and generating the boundary curves are automated, but the user supplies the necessary perception,

zoning design knowledge, and implicit bias, i.e., the aspects of the problem which are difficult to automate).

3. Increase the level of system automation *incrementally* by replacing the parts previously supplied by the user, one at a time, as described in the sections below.
4. Use existing grid generation capabilities.

In the sections which follow, pragmatic solutions to the problems of perception, design knowledge, and user bias in flow field zoning are described.

SHAPE – A MATTER OF INTERPRETATION

To automatically design a flow field zoning for a given problem, the system must have available qualitative shape and configuration information. This information may be obtained in one of two ways: interactively from the user or automatically through extensive processing of the raw geometric data input by the user. Interactive input was selected for EZGrid for the following reasons: 1) data processing can be time-consuming, even for cases which are simple and obvious to a user; 2) the shape distinctions typically resulting from such processing¹² are finer than necessary for this application; 3) the way an object is described by a user may reflect some bias, and can radically affect a zoning design; and 4) having the user describe the configuration permits the system to share the user's focus on object groupings as a way to decouple portions of the problem where possible (objects which are far apart or separated by one or more other objects may have little influence on each other in terms of how the zoning is designed). The user still provides the perceptual information, but explicitly, in a consistent manner, only for the input geometry, and only during the set-up phase at the beginning of an EZGrid run.

To make the interactive input of shape and configuration descriptions as consistent and as painless as possible, a simple shape and configuration language was developed based on Brady's hypothesis¹² that all shapes have identifiable subshapes. In EZGrid, object shapes are composed of one or more primitive parts which are described by various attributes (orientation, length, width, etc.). Each part has a front end, a back end, a top side, and a bottom side. Ends can be blunt, sharp, or base. Sides can be straight, convex, or concave. Common end/side combinations are given names, such as ellipse (both ends blunt, both sides convex), teardrop (one end blunt, one end sharp, sides any value), and bullet (one end blunt, one end base, both sides convex). Parts fit together via "joins", a simplified version of Brady's join concept. Configurations are described by grouping objects that the user feels have a direct influence on each other, and then providing nearest distances and relations among objects within a grouping and among groupings. Object grouping permits a decomposition of the problem into simpler subproblems, as is discussed in the section on encoding zoning design knowledge.

The qualitative shape and configuration information input by the user has a great impact on the design of a zoning for any given problem. An example of the effect of shape description is shown in Figures 3a-d, in which the geometry, inflow conditions, and user bias are all identical. The only difference lies in how the shape has been described qualitatively. The objects in Figures 3a-d have been described, respectively, as an ellipse, a bullet, a wedge (one end sharp, one end base), and a teardrop. The "ellipse" is surrounded by a single zonal grid with what is known as a C-type topology. A two-zone grid was designed for the "bullet" case. The "wedge" flow field was partitioned into

three zones, and the "teardrop" shape was surrounded by two zones, each with an H-type topology. The four zonings are quite different, and demonstrate the large effect that shape description has on flow field zoning design.

INCORPORATION OF USER BIAS

One of the problems associated with flow field zoning is that the experts do not agree on what makes a good zoning. One way to deal with this problem in the development of a knowledge-based system is to establish a standard, or archetypal, set of guidelines. The drawback of this approach for zoning is the possibility that the standard could be totally unacceptable to some experts, and totally acceptable to none. The approach which was chosen, and which has worked well, is to establish a *tunable* zoning archetype. A user can tune the archetype to reflect her or his own bias.

The criteria for designing and evaluating flow field zonings can be categorized as either objective or subjective. Objective criteria include basic guidelines for the type of zoning being used, such as: zones are empty and topologically four-sided, zones abut without gaps rather than overlap, zonal boundaries do not cross each other or the boundaries of the input geometry, and the outer boundary location depends on the physical conditions of the problem. Subjective criteria comprise what is commonly referred to in expert systems parlance as "standard practice." As noted above, zoning practice is not standard. These subjective criteria, which depend on a user's bias, form the basis for the tunable zoning archetype.

The bias an expert user brings to a zoning design problem involves a variety of factors: 1) the particular capabilities of the user's flow solver code, for example, how boundary conditions and singularities are handled, what sort of turbulence model is used, and what effect grid skewness has on the robustness of the code; 2) the user's own experience (often based on a specific flow solver), which determines the user's threshold of tolerance of inaccuracies caused by grid skewness, discontinuities, singularities at body surfaces, and zonal boundary intersections with body surfaces; 3) the user's objectives for the problem at hand — for example, the finest resolution is needed at body surfaces and in the wake in order to calculate drag accurately, or high grid point efficiency is needed to get the greatest accuracy with a small, fixed number of grid points; and 4) aesthetics — "I don't like the look of that discontinuity in the boundary curve."

To incorporate a user's bias into the design and evaluation processes, it must first be parameterized. The archetype is defined as the collection of parameters chosen to capture zoning user bias, and is tuned by the assignment of weights to each parameter. A list of zoning parameters and their possible weight values is found in Table I. Note that the values are all qualitative. The archetype is intended both to guide the design of zonings and to evaluate completed zonings. These qualitative values are used by the design rules to influence a zoning design, as evidenced by the results in Figures 4a-c. The geometry, inflow conditions, and shape description are identical for each case in Figure 4. Different zonings result from different archetype parameter values. The zoning in Figure 4a was generated for a user who places more importance on surface quantities (such as lift and surface drag) than on field quantities or the wake. Figure 4b results when those priorities are reversed. The zonings differ mainly in topology — an O-type topology was automatically selected for 4a and a C-type topology for 4b. The two-zone zoning in Figure 4c shows

the EZGrid method of compromising when both surface quantities and wake resolution are considered important.

Numerical values are more convenient than qualitative ones for evaluating and comparing zonings. To translate the qualitative values into numerical ones, the archetype was calibrated in the following manner. Three different configurations of NACA0012 airfoil pairs were selected as test cases: horizontally aligned, vertically aligned, and staggered. For the first two cases, three candidate zonings were generated using EZGrid in interactive mode, and four candidate zonings were generated for the staggered configuration. These three test cases were shown to five flow field zoning experts. The experts were asked first to select weights for the archetype parameters consistent with their own views and appropriate for the test cases. No two of the resulting archetypes were identical. They were then asked to order the candidate zonings for each test case by preference, presumably consistent with the archetype as they had tuned it. Their orderings for each case are represented by the first number in each column in Table II. For example, expert 1 rated zoning A as second best, zoning B as worst, and zoning C as best for the first case.

The second number in each column of Table II is the ordering given by EZGrid for the same cases. The EZGrid preferences were arrived at by comparing the scores calculated for the candidate zonings. Each parameter has a measurement function which, when applied to a zoning (prior to grid generation), yields a number that denotes a penalty for that aspect of the zoning. The raw penalties for each case did not vary from expert to expert, of course, but the weights which multiply those values come from each expert's tuned archetype, so when the results are summed and compared, the EZGrid preference in each case does vary from expert to expert. The assignment of numerical values to the qualitative weights chosen by the user was adjusted so as to maximize the number of matches between expert and EZGrid. Out of fifteen orderings, in only two does EZGrid fail to choose the same "best" candidate as the expert. It would be misleading to state that these results are statistically significant, but it is reasonable to claim, based on this study, that user bias has a measurable effect on flow field zoning design, and that the proposed zoning archetype is a promising method of evaluating zoning results in the absence of universally accepted criteria.

ENCODING ZONING DESIGN KNOWLEDGE

There are many approaches to design. The process of flow field zoning is modeled as being of the design-by-composition variety. A zoning design can be described by a composition of primitive zoning actions. To automate zoning, it is necessary either to automate the choice of an action at each stage of the design, or to follow a plan (a sequence of actions) constructed or selected at the outset.

Consider zoning design as a search problem in which zoning knowledge is used to restrict the search for an acceptable solution. The space of possible solutions (or search space) consists of all possible combinations of zoning actions. Weak heuristics are bits of knowledge that help to prune the number of possibilities, but do not narrow the range of possibilities sufficiently to eliminate the need for search. The nature of the zoning search space is such that if only weak heuristics are used (involving number of objects, prior actions, containment information, and inflow conditions), the number of possible solutions remains large, and much redundancy in the final zoning design candidates is unavoidable. If strong heuristics are

used (involving qualitative shape and configuration information, user bias, zoning design knowledge, and fluid dynamics/CFD knowledge), the search space is narrowed to one or possibly several action sequences. In fact, search is eliminated. For such a search space, the plan option was adopted.

Zoning plans are constructed by assembling predefined subplans. Subplans are sequences of zoning actions applicable to a single grouping of objects. The bulk of the zoning design knowledge in EZGrid is contained directly by these subplans. If certain preconditions about an object grouping are true, a complete subplan is asserted into the database. If the configuration is described by only one grouping, the subplan becomes the plan. If there is more than one grouping, more than one subplan is selected and assembled together to produce a plan. The rules which determine the way in which the subplans are combined contain the remainder of the zoning design knowledge. Figure 5a shows a three-zone viscous zonal grid for a single rotor blade cross-section. Its plan was composed of a single subplan. For the case of a cascade of rotor blades, represented here by just two blades, one could either develop a subplan for a grouping consisting of two vertically-aligned blades, or describe the configuration as having two groupings, each consisting of one blade. In the case of two groupings, the subplan of Figure 5a would be selected twice and combined. The five-zone grid resulting from this assembly is shown in Figure 5b. The result of a somewhat different combination of the same subplan is shown for the rotor-stator pair in Figure 5c. The final example of subplan assembly is the seven-zone zoning of Figure 2, which was generated automatically following a plan assembled from three simpler subplans. The assembly of plans from subplans increases the system's efficiency (it is not always necessary to add new subplans to handle new configurations) and generality (designs for complex problems may be composed of designs for simpler problems). This approach has proved successful for the test problems selected.

CONCLUSIONS

Flow field zoning is an effective solution to the three-dimensional grid generation problem of computational fluid dynamics. To further exploit the potential of zonal approaches, zonal grid generation must be automated. As a stepping stone to an automated three-dimensional zonal grid generation capability, a two-dimensional flow field zoning demonstration system, EZGrid, has been implemented using knowledge-based techniques. This paper focuses on those aspects of flow field zoning which make the use of such techniques challenging: the element of perception, lack of expert consensus, and the modelling of zoning as a design process. In the case of perception, the solution involves the use of a simple shape and configuration language to facilitate the interactive input of such information by the user. The lack of expert consensus is overcome not by imposing a rigid standard, but by the development of a tunable zoning archetype, which affects the zoning design and provides a means of evaluation. The design knowledge essential to the construction of a flow field zoning is encoded in the form of subplans, which, when selected, are assembled into a plan for designing and generating a zon-

ing. A design problem is thus transformed into a simpler selection and assembly problem.

It is clear, then, that the guidelines for choosing a problem to which knowledge-based techniques can be applied are not hard-and-fast rules. The successful implementation of EZGrid demonstrates that even if some aspects of a problem prevent it from being a perfect application for a knowledge-based approach, time and persistence can overcome the problems which arise.

REFERENCES

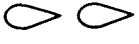


1. Kutler, P., Steger, J.L., and Bailey, F.R., "Status of Computational Fluid Dynamics in the United States," AIAA paper 87-1135-CP, AIAA 8th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 9-11, 1987.
2. Chapman, D.R., "Trends and Pacing Items in Computational Aerodynamics," *Lecture Notes in Physics*, Vol. 141, Ed. Reynolds, W.C. and McCormack, R.W., Springer-Verlag, Berlin, Heidelberg, Germany, 1981.
3. Kutler, P., "A Perspective of Theoretical and Applied Computational Fluid Dynamics," *AIAA Journal*, Vol. 23, 1985.
4. Thompson, J.F. and Steger, J.L., "Three-Dimensional Grid Generation for Complex Configurations - Recent Progress," AGARDograph No. 309, 1988, Yoshihara, H., Ed., North Atlantic Treaty Organization Advisory Group for Aerospace Research and Development.
5. Rai, M.M., "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations," AIAA Paper 84-0164, AIAA 22nd Aerospace Sciences Meeting, Reno, Nevada, Jan. 9-12, 1984.
6. Hessenius, K.A., and Rai, M.M., "Applications of a Conservative Zonal Scheme to Transient and Geometrically Complex Problems," AIAA paper 84-1532, AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, Snowmass, Colorado, June 25-27, 1984.
7. Genesereth, M.R., Greiner, R., Grinberg, M.R., and Smith, D.E., "The MRS Dictionary," Stanford Heuristic Programming Project Report No. HPP-80-24, Stanford, California, Jan. 1984.
8. Russell, S., "The Compleat Guide to MRS," Stanford University Department of Computer Science Report No. STAN-CS-85-1080, Stanford, California, June 1985.
9. Steinbrenner, J.P., "GRIDGEN2D - Interactive Elliptic Surface Grid Generation - User's Manual," General Dynamics Report No. CFD 063-4-8601, Fort Worth, Texas, June 1986.
10. Hayes-Roth, F., Waterman, D.A., and Lenat, D.B., Eds., *Building Expert Systems*, Addison-Wesley, 1983.
11. Clancey, W.J., "Classification Problem Solving," Proceedings of the AAAI National Conference on Artificial Intelligence, Austin, Texas, Aug. 1984, pp. 49-55.
12. Brady, M. and Haruo, A., "Smoothed Local Symmetries and Their Implementation," *The International Journal of Robotics Research*, Vol.3, No. 3, Fall 1984, pp. 36-61.

ORIGINAL PAGE IS
OF POOR QUALITY

Table 1 Zoning Archetype Parameters

ARCHETYPE PARAMETER	POSSIBLE VALUES
SIMPLICITY ZONE CORNER SKEWNESS ZONE SIDE SMOOTHNESS ZONE SIDE MAPPING DISPARITY GRID POINT EFFICIENCY ORTHOGONALITY AT BODY SURFACES SURFACE vs. FIELD QUANTITIES WAKE RESOLUTION	NO LOW MEDIUM HIGH } IMPORTANCE
ZONE TUPLE POINTS SINGULARITIES AT BODY SURFACES ZONE/BODY INTERSECTIONS VISCOSITY IN MORE THAN ONE DIRECTION	ALLOWED BUT NOT IMPORTANT ALLOWED SOMEWHAT DISCOURAGED DISCOURAGED STRONGLY DISCOURAGED NOT ALLOWED

Table 2 Zoning Archetype Calibration Results

GEOMETRY	CANDIDATE ZONING	CANDIDATE ORDERING BY PREFERENCE (HUMAN EXPERT/EZGRID)				
		EXPERT 1	EXPERT 2	EXPERT 3	EXPERT 4	EXPERT 5
HORIZONTAL NACA0012 PAIR 	A	2/2	1/1	2/2	1/1	1/1
	B	3/3	2/3	3/3	2/3	3/3
	C	1/1	3/2	1/1	3/2	2/2
STAGGERED NACA0012 PAIR 	A	3/3	4/4	4/4	2/2	4/4
	B	1/1	3/3	1/1	1/1	1/3
	C	4/4	2/2	3/3	4/4	3/2
	D	2/2	1/1	2/2	3/3	2/1
VERTICAL NACA0012 PAIR 	A	2/2	1/1	1/1	3/2	1/1
	B	1/1	2/2	2/2	2/1	2/2
	C	3/3	3/3	3/3	1/3	3/3

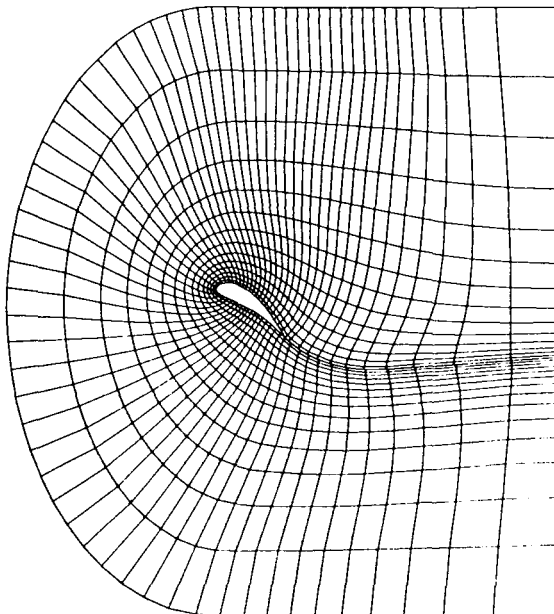


Figure 1 A single-zone grid.

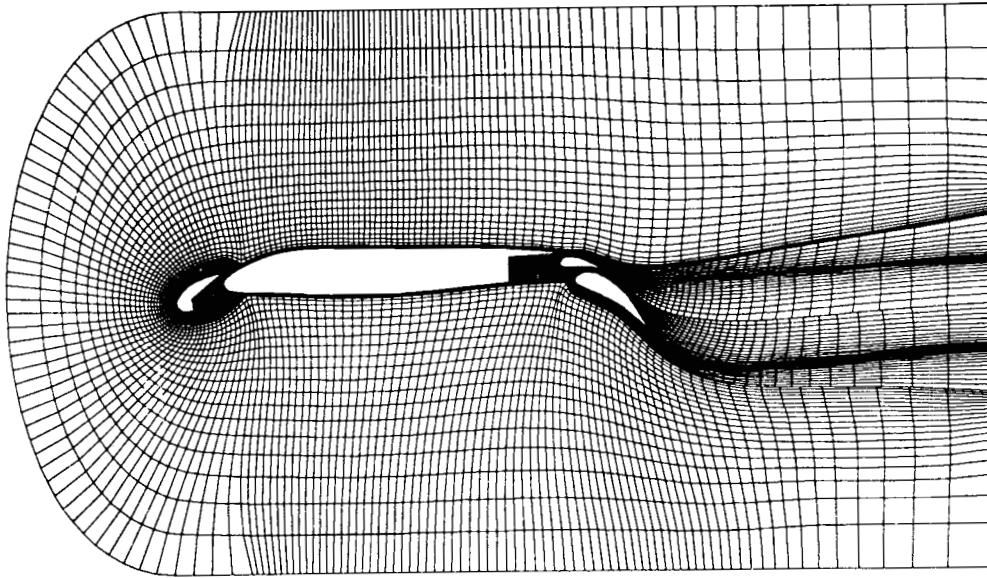
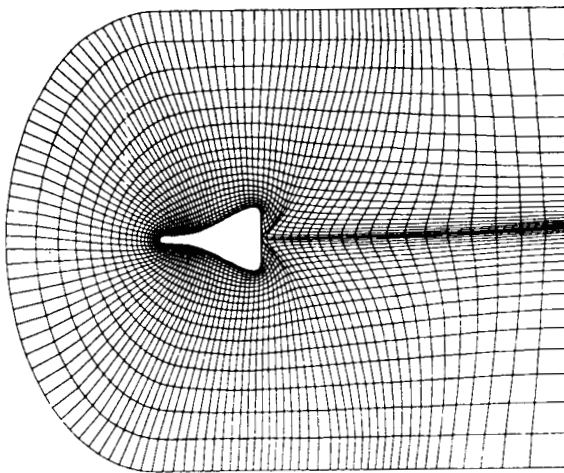
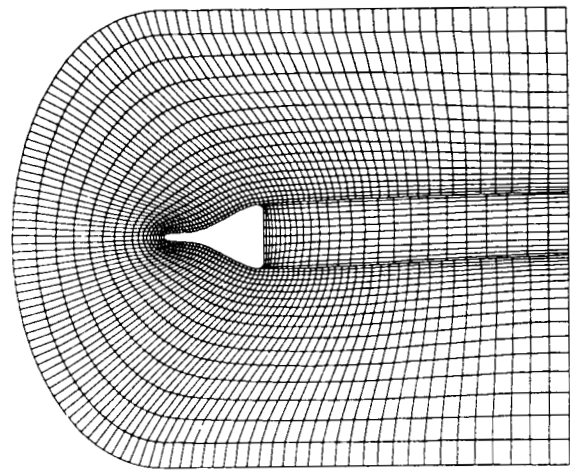


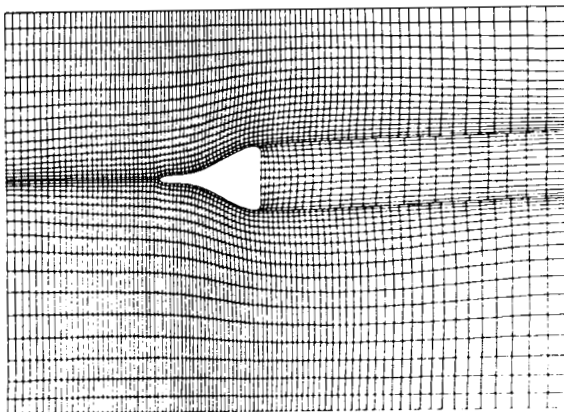
Figure 2 A 7-zone grid.



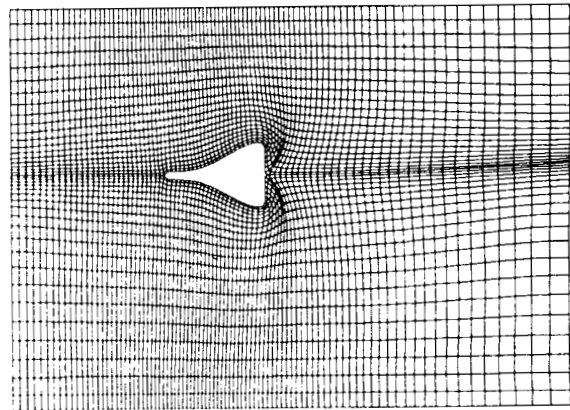
(a) ELLIPSE



(b) BULLET



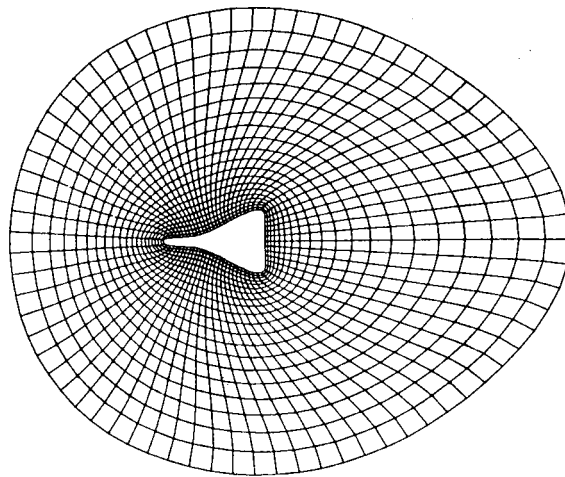
(c) WEDGE



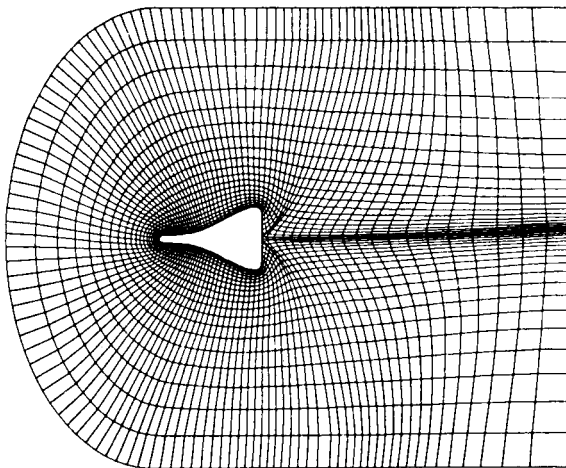
(d) TEARDROP

Figure 3 Effect of qualitative shape description.

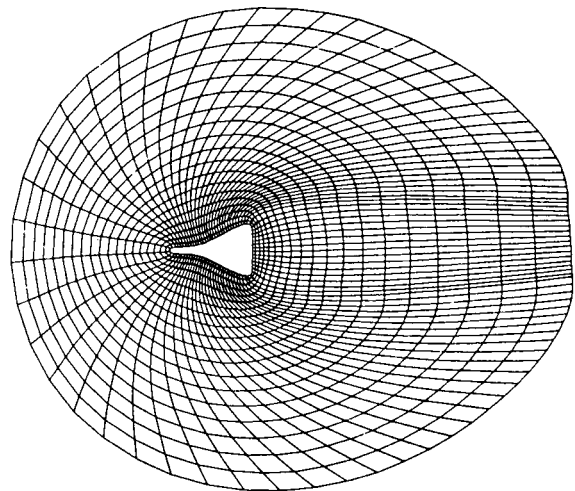
**ORIGINAL PAGE IS
OF POOR QUALITY**



**(a) WAKE – LOW IMPORTANCE
SURFACE QUANTITIES – HIGH IMPORTANCE**



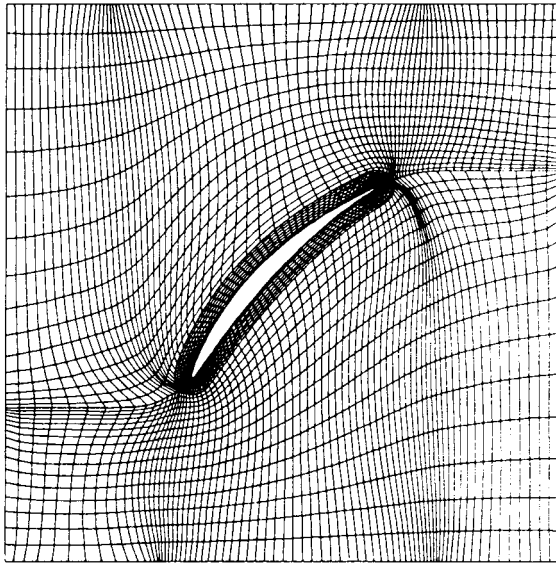
**(b) WAKE – HIGH
SURFACE QUANTITIES – LOW**



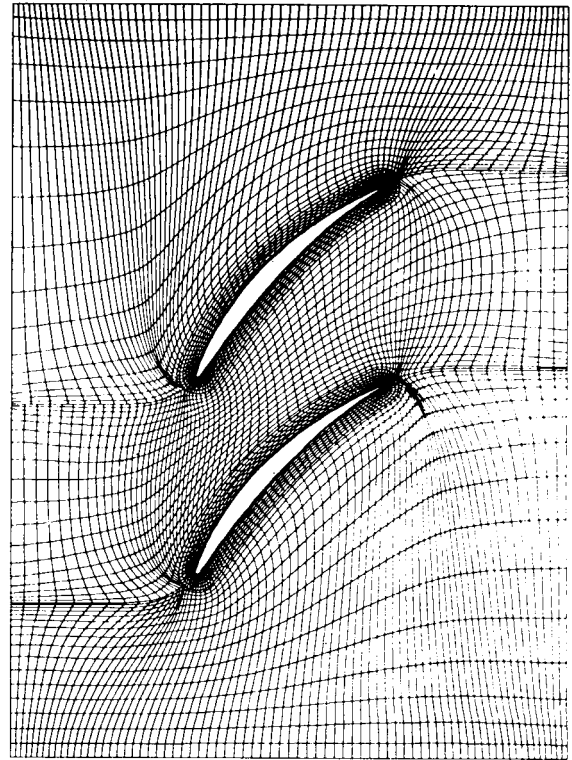
**(c) WAKE – HIGH
SURFACE QUANTITIES – HIGH**

Figure 4 Effect of user bias.

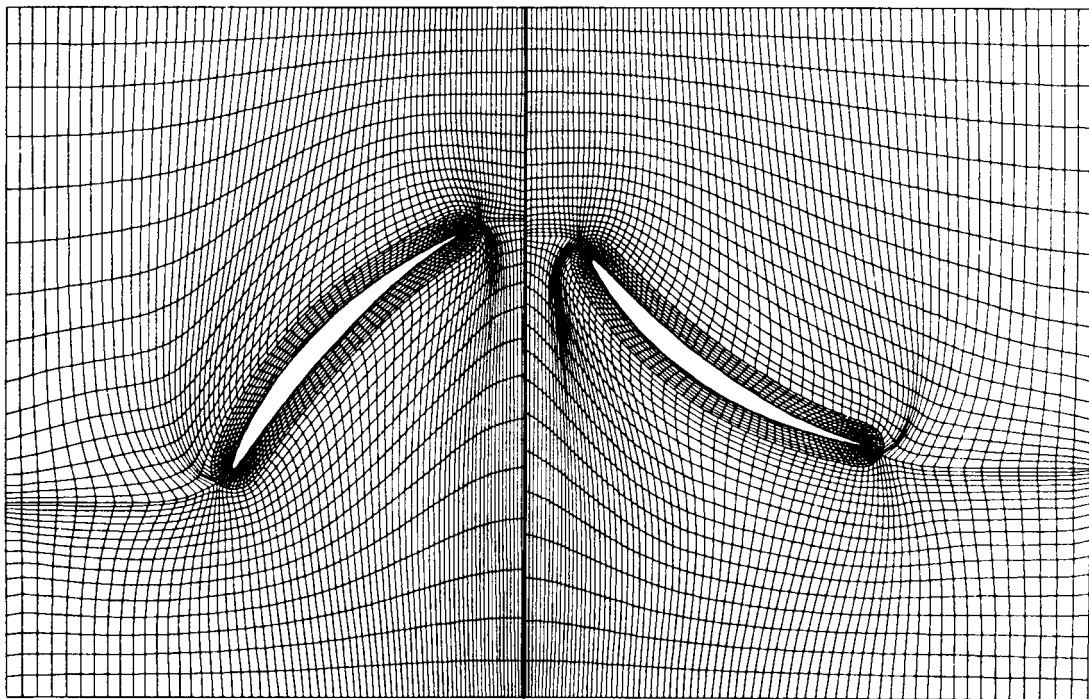
ORIGINAL PAGE IS
OF POOR QUALITY



(a) SINGLE ROTOR BLADE



(b) ROTOR CASCADE



(c) ROTOR-STATOR PAIR

Figure 5 Effect of assembling subplans.

CIRCA 2000 OPERATIONS CRITERIA

Arthur L. Scholz
Boeing Aerospace Operations
Cocoa Beach, FL 32932-0220

William J. Dickinson
NASA
Kennedy Space Center, FL 32899

Abstract

The current Shuttle Program was used as a working model and certified data source in the identification of STS operational cost drivers. Changes to flight hardware, processing methodologies, and identification of automation applications that would reduce costs were derived by reference to that data. The CIRCA 2000 Criteria were developed using these critical analyses of the on-going Shuttle Program. Several innovative suggestions are reviewed.

In 1986, the Kennedy Space Center commissioned BAO (Boeing Aerospace Operations) to perform the SGOE/T Study, (Shuttle Ground Operations Efficiencies / Technologies Study". This Study was a one year contract (Phase 1) with two priced options (Phase 2 and Phase 3). We are currently in Phase 3 of that Study. Each Phase of the Study has had a different thrust. The Phase 1 primary objective was to identify the Shuttle Program operational cost drivers and reduce the overall operational cost either through improving the efficiency of the ground operations or with the addition of selected technology elements to cut costs. The results of the study indicated that although it may be too late to "significantly" change the existing Shuttle System per se, development of launch site criteria for use by the various design agencies and their contractors would be beneficial for future programs, either manned or unmanned. One of the significant conclusions was that increased application of automation to evaluate systems and conduct operations will provide several ways of reducing launch operations costs and providing benefits:

- o Increase the speed of the total checkout (reduce time-in-flow requirements)

- o Reduce manpower requirements
- o Reduce the possibility of human error
- o Minimize documentation changes (improve test-to-test consistency) and provide the potential for "learning curve" reduction in the time required for manual tasks.

The data also clearly indicated that the lack of emphasis on maintenance requirements during the early design portion of the Program has had a very significant impact on recurring, operational costs. Bypassing these considerations in favor of other high priority items to save front-end costs in the design phase of the Shuttle has significantly increased operational costs at KSC and, therefore, Life Cycle Costs for the Program.

A Phase 2 product of the SGOE/T Study was development of the description of a generic launch vehicle to be operational by the year 2000 and titled "Circa 2000" or "Circa 2K" for short. The objective was to use the operations cost drivers identified in Phase 1 of the Study. The approach was to develop individual operational requisites for the four segments of a launch system.

The Circa 2000 System's four areas include:

Management and System Engineering

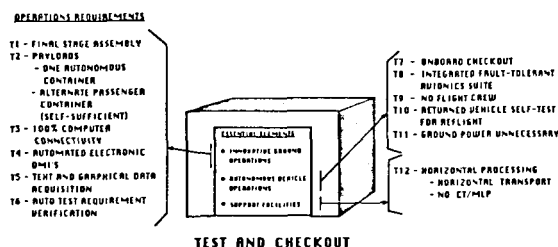
Vehicle Design

Test and Checkout

Launcher/Pad

In each of these areas, the Circa 2000 system defines design requirements that will **significantly reduce** the launch operations costs contribution to Life Cycle Costs (LCC). Personnel interested in further information are referred to William J. "Bill" Dickinson, phone number (407) 867-2780, at the Kennedy Space Center. Due to the primary interest of this audience, this paper will concentrate on the "Test and Checkout" segment (see *Figure 1*), because that area will benefit the most from the incorporation of additional automation techniques.

Figure 1
TEST AND CHECKOUT



Some of the CIRCA 2000 criteria for that segment include:

I. 100% Computer Connectivity

Operations Requirement:

All computers associated in any manner with operations, flight or ground, must maintain complete connectivity (bridging).

Rationale:

The amount of data required to support and maintain any operational system requires that efficiency in acquisition of

operational data be always maintained. Paperwork (its development, maintenance, use and control) currently requires a large portion of the operations budget. A significant reduction in the total Life Cycle Costs can be achieved by intensive application and use of automation to replace paperwork.

Sample Concept:

Utilization of commercial DBMS (Data Base Management System) which support SQL (Standard Query Language) and provide for data import and export in a manner that meets the requirements of MIL-STD-1840A.

Technology Requirement:

Distributed DBMS that will provide the required flexible computer connectivity.

II. Automated Electronic OMI's

Operations Requirement:

Operational and support procedures should be computer-based and maintained.

Rationale:

Automation of the OMI (Operations and Maintenance Instructions) process; i.e., development, maintenance, and use, provides improvements in:

- o Costs
- o Discipline of usage
- o Performance data verification
- o Configuration change compliance

Sample Concept:

Assembly and checkout procedures would be received from each vendor electronically (per MIL-STD-1840A, including graphics). These data would then be processed into an operational-site procedure format. As procedures are scheduled for performance, the test conductor would initiate them from his terminal and follow the displayed test progress. The displays will include instructions for manual operations, progress of the automated test sequences, or the requirements for hardware replacement and retest in the event of an out-of-tolerance test result indication.

Technology Requirement:

Procedure authoring and update, standardized text and graphics formats.

III. Automatic Test Requirements Verification

Operations Requirement:

Satisfaction of approved test requirements will be automatically correlated with the completion of the associated procedures. Completion of individual test requirements will be verified within the automated testing system.

Rationale:

Current manual method is labor intensive, inefficient, inadequate, and error prone.

Sample Concept:

A truly paperless, automated OMI will have sequence execution controlled by scheduling systems that track the completion of each procedure and task. As each task is completed, without error, or after maintenance and retest is accomplished, all associated test requirements would be automatically verified.

Technology Requirement:

Operational systems distributed data processing, networking, and computer/data connectivity.

IV. Integrated Fault Tolerant Avionics Suite (IFTAS)

Operations Requirement:

Avionics systems must provide for higher reliability by providing several levels of fault tolerance thru redundancy to support mandated system availability.

Rationale:

To support on-board checkout and mission success, the entire avionics suite must be designed to provide that level of fault tolerance required to assure that the system is available when required. This is best accomplished by assuring the

robustness of all mission critical systems, and providing fault tolerance where it is required (similar to the minimum equipment list used by commercial aircraft).

Sample Concept:

Future systems must be designed such that systems in general can be dynamically configured to provide for more than one function. Should an allocated processor or sub-system fail, another processor with a lesser priority function should be assigned to reconfigure and perform the function of the failed processor. This will force a high degree of commonality, require distributed processing, and provide a system more capable of surviving in adversity.

Technology Requirement:

Distributed processing, development of layered architectures, commonality of equipment elements.

V. Returned Vehicle Self-Test for Reflight

Operations Requirement:

After flight, the returned vehicle should have sufficient self-test capability to verify its flight readiness or provide problem isolation down to the LRU (Line Replaceable Unit).

Rationale:

To accomplish an order-of-magnitude cost reduction, we must strive to achieve the 160-hrs. (or better) turnaround time. The original STS turnaround objective was 160-hrs. STS actual processing times have grown an order of magnitude beyond that original goal.

Sample Concept:

During flight, BIT identifies/records anomalies. After landing, BIT/BITE isolates problem to LRU level. After replacement, BIT retests the system and verifies flight readiness.

Technology Requirement:

Development of BIT/BITE to meet specific requirements.

VI. Autonomous Guidance Navigation and Control (GN&C)

Operations Requirement:

Eliminate vehicle dependence on GSE for test and checkout.

Rationale:

Onboard BIT/BITE of GN&C can eliminate, simplify, or reduce the requirement for ground support operations.

Sample Concept:

Boeing 757/767 or advanced military aircraft computerized electronics providing self-test and fault identification with fault-tolerant computers. Ability to replace circuit boards without system shutdown. Easy accessibility.

VII. Software Commonality

Operations Requirement:

The vehicle should utilize the same set of software for ground operation test, integration, and for flight.

Rationale:

Current STS ground operations are accomplished with several different programs depending on the stage of testing. This results in many hours of time for reloading the main computer memory. For example, the final prelaunch load requires 14 clock-hours to accomplish.

Sample Concept:

The Avionics should be designed as a distributed system with one or more high speed buses providing communications between subsystems as required.

Each subsystem should have the capability of autonomous ground operations by commanding the system to a standalone mode.

In this mode all required external stimuli would be simulated by the subsystem in a sufficient manner to verify its proper operation. This would allow each subsystem to be tested independently of the operational state of the other systems. When all ground testing and vehicle integration is complete, each subsystem would be commanded to the flight mode without additional computer reloading.

Technology Requirement:

Distributed, layered architecture.

In a paper presented last year (1987) at the Space Congress in Cocoa Beach, FL, David Lowry and Tom Feaster described plans for research and development of computerized tools to incorporate supportability factors in the early phases of system design (Reference 2). The concept includes the role that CAE/CAD/CAM should play in improving design for supportability. There is an accumulation of evidence from recent research performed by the Human Resources Laboratory at Wright-Patterson Air Force Base that indicates that the accumulation of operational maintenance and logistics support characteristics must begin early in the development of system concept studies. This research indicates, also, that one of the best ways to improve design for support is to include the operational maintenance and logistics data and factors directly in the daily working procedures and systems used by the design engineering personnel (Ref. 3).

There is a need, then, to develop the technical capability to develop generic operational maintenance criteria, logistics factors and provide operational requirements and criteria feedback directly into the CAD process being used by the aerospace industry. This technical capability does not exist today except in limited scope and then only in isolated cases.

The current status of design for support is primarily that of studies and analyses being performed "off-line" from the main "performance-oriented" engineering design activities; usually being performed "after the fact" without any real input to major design

decisions. The development of the technical capability to enter maintenance and logistics factors (along with equipment design information and performance parameters) directly into the main CAD process can change this picture.

Equipment and systems designed from the beginning with accessibility and supportability to meet operational criteria can become a standard, on-line design product.

Computer-based, automated analysis models are an essential part of the CAD process. Presently these models are used to assess system element performance characteristics as well as weight & balance. These automated analysis models are one of the reasons for the quick reaction time of the CAD process. Consideration of operations maintenance and logistics data in the analyses models will also be necessary if meeting approved Life Cycle Cost targets is going to truly be a primary objective in the future.

The ability to view objects in three dimensions is now resident within many CAD systems. Color presentation of objects is now possible. These characteristics will afford opportunities to use a totally integrated CAE/CAD/CAM system to perform maintainability evaluations of proposed equipment during early design without the high cost of Class 1 mockups.

The design and drawing data generated by CAD can be, and are, being bridged to the databases that operate the numerical controlled machines within the manufacturing facility. The data flows from CAD to CAM and eventually, by hardcopy, to field and service organizations. Unfortunately, the databases that are used in operational maintenance and logistics analysis models have not been linked with the CAD/CAM engineering databases.

Design tasks in the future will require interchange of operational maintenance, accessibility, and supportability criteria and data with the established CAD/CAM systems if costs are to be truly optimized and reduced to meet the current cost targets.

CHALLENGE

REDUCE LIFE CYCLE COSTS

BE CREATIVE ! !

BE INNOVATIVE ! !

DESIGN the SUPPORT
not
SUPPORT the DESIGN

References

1. Shuttle Ground Operations Efficiencies /Technologies Study, Phase 2 Final Report, Vol. 6, May, 1988, Kennedy Space Center.
Contacts: A. L. Scholz, Boeing;
W. J. Dickinson, NASA-KSC
2. Lowry, David J., Boeing; Feaster, Thomas A., NASA-KSC; "Regaining Space Leadership Through Control of Life Cycle Costs", Proceedings of the Twenty Fourth Space Congress, 1987.
3. Askern, William B., "Maintenance and Logistics Factors in Computer Aided Design", Air Force Human Resources Laboratory, Wright Patterson AFB, NASA RECON 84A16548.

ARTIFICIAL INTELLIGENT DECISION SUPPORT FOR LOW-COST LAUNCH VEHICLE INTEGRATED MISSION OPERATIONS

Dr. Gerard P. Szatkowski
General Dynamics / Space Systems Division
5001 Kearny Villa Rd., San Diego, Ca. 92138

Dr. Roger Schultz
Abacus Programming Corporation
14545 Victory Blvd., Van Nuys, Ca. 91411

Sponsored by: Flight Dynamics Lab., AFWAL WPAFB

ABSTRACT

This paper reviews progress in a current study assessing the feasibility, benefits, and risks associated with AI Expert Systems applied to low cost space expendable launch vehicle systems.

This study is in support of the joint USAF/NASA effort to define the next generation of a heavy-lift Advanced Launch System (ALS) which will provide economical and routine access to space. The significant technical goals of the ALS program includes: a 10 fold reduction in cost per pound to orbit, launch processing in under 3 weeks, and higher reliability & safety standards than current expendables.

General Dynamics with Abacus, under the auspice of Wright Aeronautical Labs WPAFB, are exploring the use of knowledge-based system techniques. This is for the purpose of automating decision support processes in on-board and ground systems for pre-launch checkout and in-flight operations. Issues such as: satisfying real-time requirements, providing safety validation, hardware & DBMS interfacing, system synergistic effects, human interfaces, and ease of maintainability, have an effect on the viability of expert systems as a useful tool.

INTRODUCTION

The Problem — We recognize that our nation's current suite of launch vehicle systems has a number of problems making them inadequate for the projected needs after the mid-1990's. High costs of above \$2,000/lb of payload delivery, a low reliability, poor resiliency (standdowns of many months for current expendables), and limited launch rate capacity are reasons behind

the joint USAF/NASA effort for an operational ALS and Shuttle II. These will serve the commercial and DoD mission models beginning in 1995. If we are to meet the goals of \$300/lb and launch rates as high as 50 missions annually, these systems and their associated ground operations segment must be made as autonomous as possible, while at the same time improving reliability and safety. This study explores the use of knowledge-based system (KBS) techniques for the purpose of automating the decision processes of these vehicles and all phases of the ground operations segment by assessing the feasibility, benefits, and risks involved.

An expert decision aid is a software approach to solving particular problems that are constantly changing over time and are complex or adaptive in behavior, the opposite of an analytical problem that is basically deterministic. Examples of these types of problems are: the re-scheduling of a vehicle checkout due to a damaged cable; or, determining if a system is indeed faulty given conflicting sensor readings. These heuristic problems require a depth of knowledge and experience (art rather than science) to form solutions quickly. Expert systems embody that collection of knowledge and experience in modular pieces that are rules and facts that describe the proper thought process for a given set of circumstances arrived at by any path. It is this modular independence that makes expert systems attractive. The incremental improvement of knowledge and experience can be built and tested readily without re-testing the rest of the software system, unlike conventional software that is difficult to maintain in a day to day changing environment.

Project Scope — The scope of the Space Transportation Expert System Study, (STRESS) is extensive. Virtually all on-board management decision functions are included as is the ground segment from pre-mission planning, through checkout and launch services, and post-flight analysis. This effort is then to define viable Vehicle Mission Management System (VMMS) architectures (and their corresponding ground system) which can evolve with this new KBS

technology, determine the most fruitful to pursue, and for specific areas derive the avenues in need of concentration. This process is depicted in Figure 1. It is clear that the focus of our efforts must be on the AI techniques supported with the knowledge of the functional systems requirements. Figure 2. shows a rough overview of the ALS program major milestones and the expected implementation of KBS as a technology.

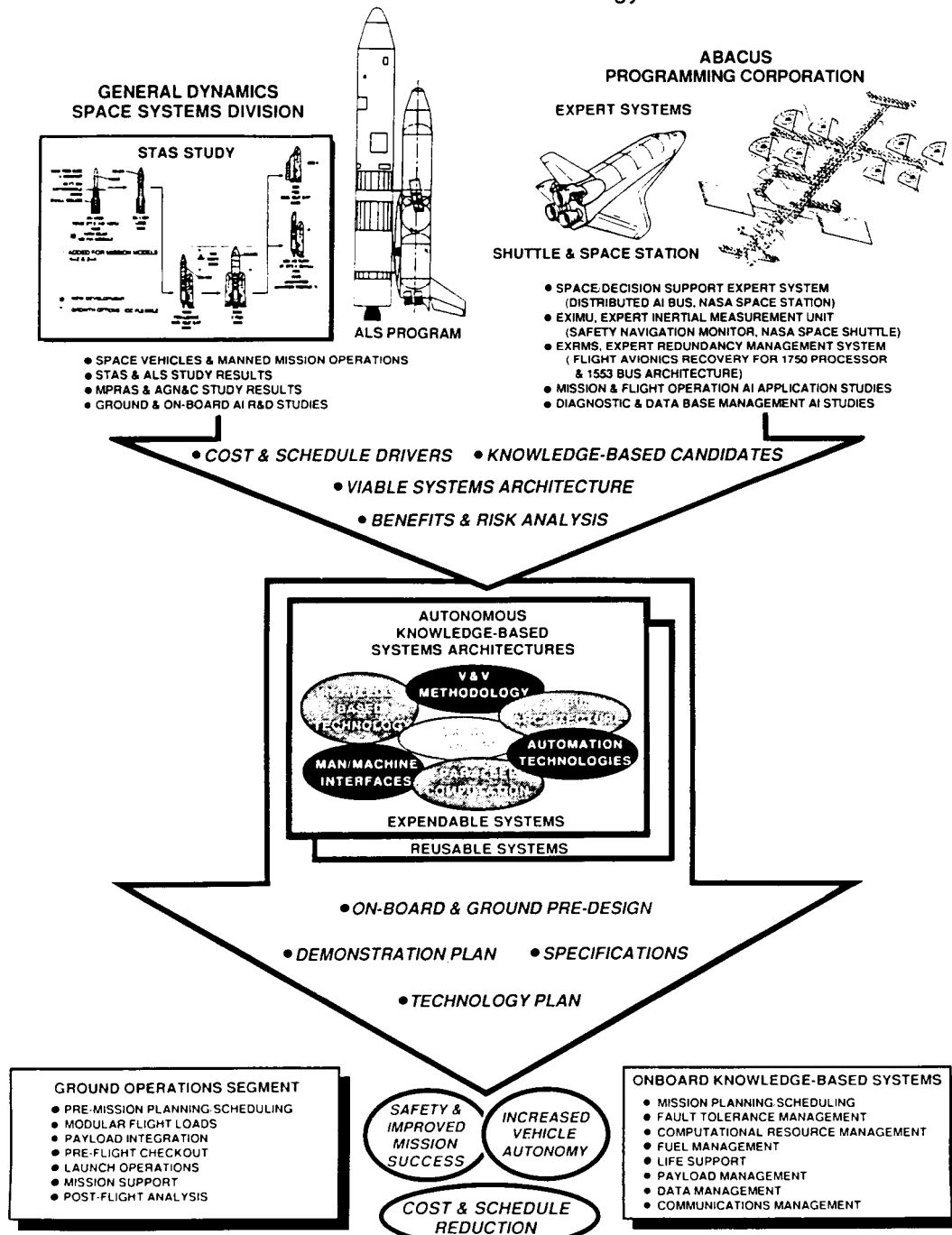


Figure 1. The Space Transportation Expert System Study, STRESS

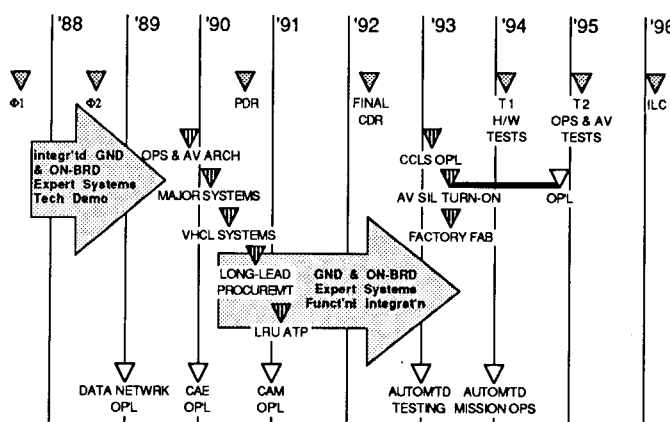


Figure 2. ALS Program Expert System Integration

Problems related to AI applications we will assess include:

- The increasing cost of failure, which leads to a high demand for success and hence system reliability
 - Ever increasing system demands, i.e., reconfigurable avionics, adaptable GN&C, heavy instrumentation, and autonomous procedures like rendezvous and docking, etc.
 - Multiple payloads in heavy lift vehicles which tend to complicate integration, planning requirements, and V&V of flight loads, and
 - Increasing flight rates, which tend to magnify the logistical dependence on ground support;
- Problems related to AI applications we will access include:

- Speed, where the current technology is not well suited to real-time situations
- V&V methodology, which has been inadequately developed to date
- Large data storage requirements, which complicates redundancy
- Knowledge representation and manipulation techniques where, to date, no standard exists and where optimized hardware has been slow to develop

To attack these problems, we must rely on technological innovations. Based on our studies, we believe the key to viable solutions must address improving speed and modularization of the multiple support functions which comprise the system. To accomplish this, two approaches we are considering are: the utilization of parallel processor techniques with a tiered management approach for modularized expert systems; and utilization of a distributed AI bus, developed for the space station, to network the functional systems. The basis of our approach is to conceptualize systems employing varying combinations of these philosophies.

BENEFITS AND LIMITATIONS OF KBS

Although there are many considerations that will be made in the application of KBS to Launch Systems, some of the major issues are:

Benefits—

- The major benefit of knowledge based languages and techniques is the obvious one that they allow the modeling and solution of problems that are inordinately difficult using conventional procedural programming.
- Ease of maintenance results from knowledge being expressed in small independent units without the complex interrelationships that result from the sequence-specific nature of procedural code.
- Ease of top-down representation and development also results from having knowledge expressed in independent units. Often it is possible to define major aspects of a problem at a relatively abstract level and gradually elaborate the knowledge to deal with more and more specific issues, in the same way an expert develops expertise. Knowledge developed in this way can be much easier to understand and verify than procedural knowledge.
- A corollary of the above advantages is that systems can be built to perform reasonably in combinations of circumstances that are unanticipated by the designer. However, there is the corresponding danger that a system will act when it should not.

Limitations—

- Real-time software applications face all the same challenges as other software applications, and in addition are faced with the problem of synchronizing with real world events. Developing a knowledge based system for real-time performance is a challenge best met by designing the performance in from the start, although tuning and hardware upgrades can have a real impact in certain cases. Design for real-time should follow two principles: 1) locate processing nodes to limit data transfer to the most essential data to avoid bottlenecks; and 2) consider distributing or pipe line processing over a number of processors.
- There is a shortage of general and proved methodologies for dealing with knowledge processing. This means that systems can be produced that are understood only by the specialists that built them and formally

unverifiable. For this reason, a concentrated effort on developing methodologies for identifying, implementing and verifying knowledge based systems, is essential.

- So far, there has been a lack of reasoning from first principles. This is inevitable, considering the immense body of knowledge that would have to be assembled to make up even a small portion of an engineer's "common sense." As a consequence, knowledge based systems cannot extend their knowledge to new fields; conversely, knowledge based systems don't know what they don't know, or when they go beyond the limit of applicability of their knowledge.
- The technology is new enough that it is still in the phase of a multitude of incompatible and immature competing tools, pending standardization (e.g., ANSI standard knowledge representations), and an industry shakeout. While the tools available can be of real value in appropriate cases, unless one is careful, systems can be developed that are not compatible with knowledge sources, and which have no migration path for upgrade or transformation to other representations.
- Typically the most sophisticated knowledge based techniques make inordinate demands on computing resources, requiring hardware such as LISP machine or multiprocessor networks. In a restricted hardware environment such as a spacecraft, it may only be possible to utilize a subset of the available techniques.

Anticipated Savings — The mission control system (MCS) of today consists of seven major task areas. The areas are flight planning, data load preparation, payload integration, training and simulation, flight control, communications and tracking, and post flight analysis. This and other studies point out many ways in which cost reductions can be achieved through the implementation of decision support systems, automated software production, and advanced information processing in the MCS areas.

Anticipated Risks — Expert system implementation introduces new risks which must be carefully managed. Some of the identified risks are:

- Problems may occur with verification and validation of the knowledge processing
- Automation might impact negatively on manual intervention, degrading safety
- Error propagation is more serious with KB systems which reduce data to conclusions

- Data security and system vulnerability problems
- Symbolic processing power required may exceed that which is available
- The problem may not be amenable to subdivision
- The cost of implementation may approach the level of anticipated savings
- The nature of the problem and knowledge required may change
- Minimal availability of top-level AI experts
- A possibly diffuse and poorly understood knowledge base

It should be emphasized that the verification and validation (V&V) of AI software is no simple task. Traditional software development methodologies use some form of path testing, measuring test coverage with metrics such as testing all branches of code. This testing often takes over 30% of the program effort. The concept of path testing does not mesh with AI systems well, because their static structure does not directly map onto their execution pattern as does traditional procedural code. For this reason the DoD software development standard is not yet applicable to the development and implementation of expert systems. It is perhaps obvious that an infinite number of tests is impractical, but the alternative is to release a program that may behave in ways not predicted by the designer. This is an issue that remains to be resolved. Our system integration laboratory (SIL) approach will be extremely useful in attacking the V&V problem, as imbedded systems become more intelligent. It is specifically aimed at hardware and software system integration.

The Abacus Expert System V&V Methodology is also a major risk reduction factor:

- Validate the inference engine separately from the knowledge base
- Utilize selected expert system forms
- Maintain separate thoroughly validated tools
- Maintain a library of test cases and scenarios
- Use an independent panel of experts for system performance validation

AI TECHNIQUES

Knowledge-Based Representation— Although in certain cases performance or hardware compatibility may be the deciding factor, knowledge representation (KR) is generally the major issue determining the success of a

knowledge based program. A successful KR must deal with the multifaceted knowledge that human experts bring to bear on problems, must be defined in terms of knowledge obtained from real sources, must be appropriate for the tools available, and must be defined with enough formality that it can be verified and validated without recourse to exhaustive testing. The knowledge engineer needs a toolbox containing a number of KR models (e.g., forward- and backward-chaining production systems, frames or schema, inheritance, escape to PROLOG, LISP, or procedural languages) which can be interlinked through knowledge gateways and/or blackboarding on standardized KRs. The Abacus AI Bus is a standardized knowledge gateway intended to coordinate distributed knowledge based processing systems.

Knowledge-Based Problem Solving — The techniques used for solving knowledge based problems cannot be separated from the KRs used to define the problems; solution techniques and representation techniques feed back and limit each other, and are both conditioned by the set of available tools. The basic issue is search, and the best techniques are those that reasonably narrow the search space most quickly, based upon intelligent use of knowledge rather than brute force. However, it is the problem KR that should drive the choice of techniques, subject only to a tool's ability to support them. What is important in a family of tools is the ability to link together a number of techniques where each technique is used where it is most appropriate. In situations where a system offering limited processing power is considered, a critical issue is the compatibility of the representations, for two reasons. First is the obvious requirement that the knowledge produced by one subsystem should be suitable for incorporation into another. Second, the behavior of the overall distributed system is difficult to verify if each element obeys a basic logic unrelated to the others.

Tool Evolution — Knowledge processing technology has advanced remarkably in the past decade, but is still relatively immature. In the next decade, tools will bring many of the current shortcomings up to a higher standard, as well as incorporating new features. The progress that is of most importance is not the latest research developments, however, but rather the standardization and certification of the already-existing technology so that it can be confidently

utilized in more critical situations:

- Tools will aid in capturing and verifying knowledge:
 - 1) Built-in domain knowledge of selected application areas will require only add the specific parameters of the problem (i.e. electronic diagnostic systems already know how to interpret a circuit diagram).
 - 2) High-quality debugging tools to view a knowledge base in a useful graphic form and check it for completeness and consistency.
 - 3) Mixed representations allow portions of a problem to be expressed in different ways, for example a slot in a frame taking a value from the operation of a rule or from a database.
 - 4) Bridges to the outside world and data sources, such as databases, spreadsheets, and communications.
- Knowledge representation will be standardized and processors certified; analogous to the development of ANSI or DoD standards for programming languages, and the certification of compilers. In this way, knowledge bases conforming to a standard representation (i.e. forward-chaining production systems) will be portable from one tool to another in the way that FORTRAN is transported from one compiler to another.
- Routine use in real-time applications will come from improved hardware, tools that are optimized to certain hardware, and the progressive identification and elimination of bottlenecks as the commonalities in common knowledge based applications become more well known.

TECHNICAL APPROACH

The STRESS Program follows a progressive plan of development:

- **Task I** — Currently in progress, we have assessed key drivers to cost, schedule, safety, and mission success by filtering inputs from the STAS, MPRAS, and AGN&C studies; and are now deriving KBS candidates with a cursory analysis of V&V effects on these drivers.
- **Task II** — Develop a technology plan so as to identify the critical performance areas and propose a research schedule to support advanced vehicle development.
- **Task III** — System partitioning predesign and tradeoffs will determine a maximum degree of autonomy that is performance and cost

effective. Evaluating the viable architectures as to benefits and risks will determine resultant effects to the onboard and ground systems requirements. Establish system specifications which will encompass the range of options identified in the predesign for each system and develop a feasibility demonstration plan that would be most representative of the critical functions and interfaces.

- **Task IV** — Develop pathfinder demonstration with functional integration to conventional avionics systems to show merit in an on-board environment.
- **Task V** — Prepare the documentation covering each of the above tasks, submit a final report for AF review, and present a final briefing of the study results.

STRESS Study Goals — The following goals are used as a point of departure:

- Decreased costs achieved with:
 - Increased autonomy, to minimize ground support time and personnel
 - Improvements in the methods used today
 - Minimized Mission Control Support by automation and vehicle autonomy
 - Reduced post-flight analysis through on-board fault logging and testing
 - Development of a standardized payload interface
- Schedule Compression will result from:
 - Availability allowing increased launch rates
 - Deferred maintenance allowed by expert system redundancy management
 - Autonomous vehicle approach, allowing automated checkout
- Improved Mission Success will result from:
 - Adaptive reconfigurable systems and control
 - Improved depth of verification/checkout:
 1. Failure prediction by data trend analysis
 2. Improved data collection and correlation
- Increased Flight Safety will result from:
 - Human operator cross-check and backup
 - Failure prediction allowed by trend analysis
 - Hierarchical end-effect failure checking
 - Replacement of men in hazardous operations

TOTAL SYSTEM DESIGN SOLUTIONS

On-Board/Ground Partitioning — The design goal is to determine the maximum degree of autonomy that could be delegated to the on-board Vehicle and Mission Management system, and establish the requirements for the complementary ground support system.

The structure of the VMMS will be based on the Pave Pillar architecture, figure 3 (without shaded interfaces), as modified by two complementary studies - the Autonomous Guidance, Navigation and Control (AGN&C) study and the Multipath Redundant Avionics Study (MPRAS). These study results, which are considering alternatives such as portioned vehicle autonomy, integrated smart sensors, and power/thermal/time optimization, will be incorporated into an overall system approach.

Maximum Degree of Autonomy — Studies to date have indicated that one of the most beneficial approaches which can be taken to reduce cost of launching vehicles is to strive for the maximum degree of autonomy for the on-board systems. The rationale is obvious; the reduction of ground support personnel team required for a vehicle system saves money in a direct and understood manner. The airborne functions considered include:

- Flight control reconfiguration
- Fault tolerance management, including self diagnostics, fault - detection, identification and prediction
- Man/Machine interfaces
- Information gathering and management
- Fuel management
- Payload management
- Mission planning, scheduling, replanning, and rescheduling
- Computational resources management
- Space/Space and Space/Ground telemetry data processing management
- Performance management
- Guidance and Flight control management

The "AI BUS": an Integrating Architecture

— The AI Bus is an architecture developed by ABACUS to provide a highly flexible structure for integration of multiple expert systems and conventional systems hosted on a mix of machines in a distributed network. The AI Bus is not a physical bus, but instead a logical one that provides standard utilities and services through standard interfaces to both knowledge based and conventional software systems. This approach allows growth towards the increased use of expert systems by providing a common framework at initial design time, along with high technology transparency for attached components. This permits substitution of components, such as a nodal parallel processor, later in the design cycle than a more conventional architecture, and it mitigates the

ORIGINAL PAGE IS
OF POOR QUALITY

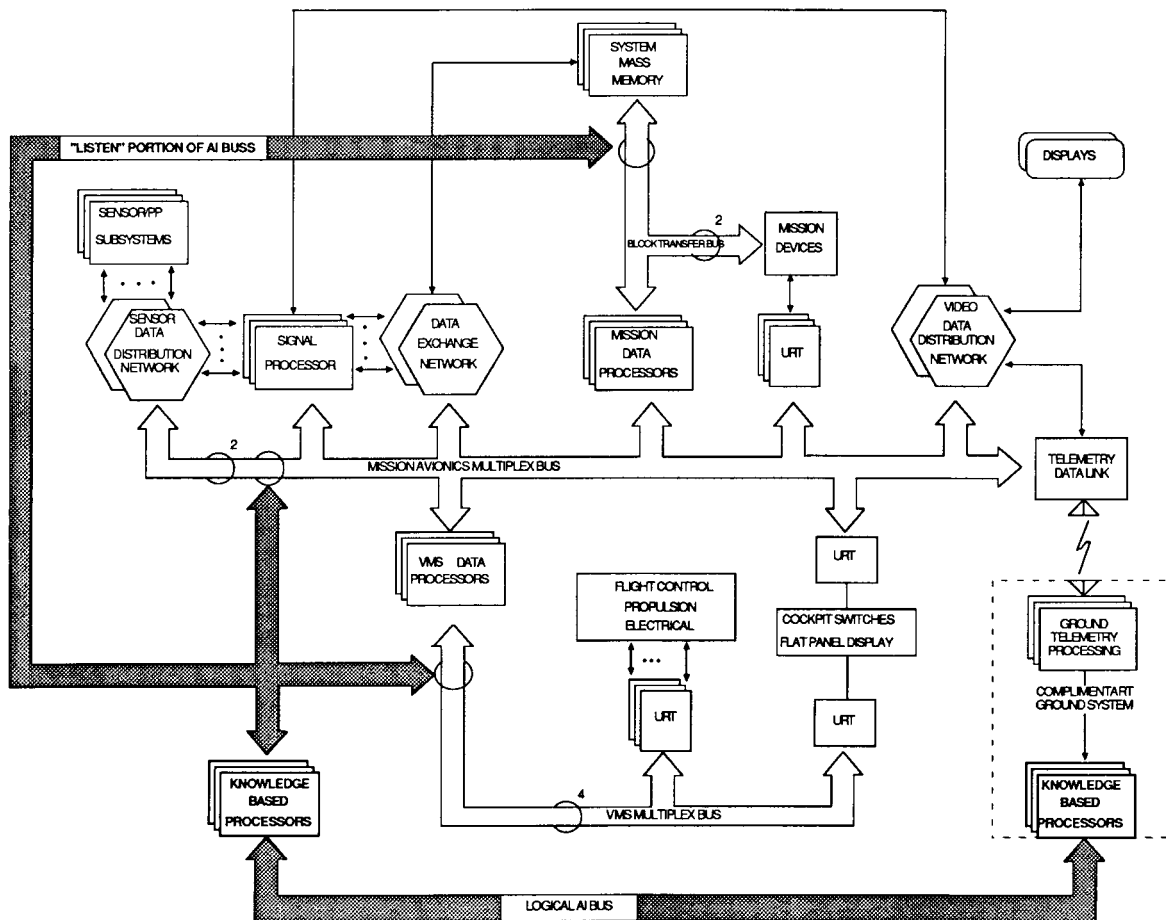


Figure 3. An AI Bus concept will provide inherent integration of knowledge based systems.

schedule risk issue. The bus is a layered architecture implemented upon the distributed data network of the vehicle avionics systems. The layers extend at the higher layers to the complementary ground based system.

Figure 3 (with shaded interfaces) shows the primary components of the AI Bus as it could be applied to the Pave Pillar architecture after it has been structured for the VMMS function. It includes:

- Common code for inference engines
- A knowledge based management system for handling working memory, truth maintenance, various knowledge representation techniques and standard database type access
- Invocation scheduling based on system events and access to the event information
- Access to system global information
- A shared, distributed blackboard structures for knowledge based systems

The AI Bus differs from conventional knowledge based systems frameworks in that it:

- Provides a "listen" capability for knowledge based systems to monitor other systems
- Is designed to operate in, and take advantage of, a distributed architecture
- Is designed to handle real time as well as conventional, consultive type knowledge based systems
- Handles a hybrid combination of rule based and conventional procedural programs

CONCLUSIONS

Experience from our launch vehicle programs and other studies show that there are many opportunities in operations that reduce costs and improve autonomy, including:

- Ground operations: daily planning support and timely work-around decisions aids
- Ground checkout: autonomous procedural operations and control, standard trends, and red-line monitoring
- On-board systems: monitoring, integration, and control
- Launch day: fly with fault diagnostics and

decision aids

- Post-flight: data reduction and analysis

This program will assess these benefits vs implementation risks and demonstrate key performance requirements to show feasibility. Development of an overall integrated architecture will be important in providing a context and focus for the follow-on demonstration prototypes, and assure the synergy of their gains in both development and use in vehicle operations.

Phase II, beginning 3Q '88, will develop, demonstrate, and validate the cost reductions predicted by the use of expert decision aids in areas that would improve ground and on-board system autonomy. Proof of concept demonstrations will be selected in order to

reduce the risk of commitment to this new technology. Each demo will be of a fractional scale; sufficient to give a good performance correlation to a full-scale implementation. These demonstrations will incrementally show:

- Ease of human interface for maintainability
- Real-time system performance
- Integration to vehicle/ground hardware, and data systems
- Validation methods consistent for ground and on-board applications.

Integration to the other related technology projects is essential to this approach. Figure 4 shows the flow of requirements, analyses, tool sets, standards, and interfaces between them. Final validation of these cost reductions will be done through demonstrations integrated into a "hot bench" environment to be established at NASA/KSC.

INTER-RELATED PROJECTS		EXPERT SYSTEMS FOR DECISION AIDS
INTEGRATED HEALTH MONITORING IHM	← →	REAL-TIME PERFORMANCE DATA TO SIZE DIAGNOSTIC & DATA COMPRESSION TECHNIQUES DEMO IN OIL MULTIPLE COOPERATING HEALTH MANAGEMENT DECISION AIDS
MULTI-PATH REDUNDANT AVIONICS MPRAS	← →	REQUIREMENTS & ANALYSIS FOR DATA & KNOWLEDGE BUS TRAFFIC FOR ON-BOARD INTERFACE TO ALS SYSTEMS INTEGRATION LAB (SIL) FOR INTEGRATED CHECKOUT
ADAPTIVE GUIDANCE, NAV, & CONTROL AGN&C	← →	STANDARDS FOR H/W, S/W, INTERFACE's, TOOL SETS, & VERIFICATION/VALIDATION REQUIREMENTS REQUIREMENTS FOR FLIGHT CONTROL ANALYSIS & TRAJECTORY DESIGN TOOLS
ADVANCED MISSION OPERATIONS	← →	STANDARDS FOR H/W, S/W, INTERFACE's, TOOL SETS, & VERIFICATION/VALIDATION REQUIREMENTS REQUIREMENTS FOR LABOR REDUCTIONS ON PLANNING, CHECKOUT, & DATA TRACKING
AUTOMATED GROUND PROCESSING	← →	STANDARDS FOR H/W, S/W, INTERFACE's, TOOL SETS, & VERIFICATION/VALIDATION REQUIREMENTS REQUIREMENTS FOR DATA ANALYSIS, AUTOMATED CHECKOUT AND INTERFACES
OPERATIONS ENHANCEMENT CENTER OEC	← →	PROTOTYPE OPERATIONS, INTERFACE STANDARDS, INTELLIGENT CHECKOUT, AND TRAFFIC REQUIREMENTS OPERABILITY, MAINTAINABILITY REQUIREMENTS
NASA/KSC	← →	PROTOTYPE OPERATIONS, INTERFACE STANDARDS, INTELLIGENT CHECKOUT, AND TRAFFIC REQUIREMENTS PERFORMANCE DATA FOR AUTOMATED CHECKOUT AND INTERFACE CONSTRAINTS

Figure 4. The information flow between these interrelated technologies is essential to our approach.

A DESIGN AND IMPLEMENTATION METHODOLOGY FOR DIAGNOSTIC SYSTEMS

Linda J. F. Williams
McDonnell Douglas
16055 Space Center Blvd.
Houston, TX 77062

ABSTRACT

This paper presents a methodology for design and implementation of diagnostic systems. Also discussed are the advantages of embedding a diagnostic system in a host system environment. The methodology utilizes an architecture for diagnostic system development that is hierarchical and makes use of object-oriented representation techniques. Additionally, qualitative models are used to describe the host system components and their behavior. The methodology architecture includes a diagnostic engine that utilizes a combination of heuristic knowledge, causal knowledge and system structure knowledge to control the sequence of diagnostic reasoning.

The methodology provides an integrated approach to development of diagnostic system requirements that is more rigorous than standard systems engineering techniques. The advantages of using this methodology during various lifecycle phases of the host systems (e.g. National Aerospace Plane (NASP)) include: the capability to analyze diagnostic instrumentation requirements during the host system design phase, a ready software architecture for implementation of diagnostics in the host system, and the opportunity to analyze instrumentation for failure coverage in safety critical host system operations.

1.0 INTRODUCTION

Space transportation systems are among the most advanced and complex systems being designed today and provide a wide variety of physical systems that will require some measure of intelligent automation. The increased complexity of these types of systems has led to extremely complex operations in manned systems that will

continue to overburden flightcrews. There is an inherent need to assist crew personnel in complex system operations through intelligent automation. In the case of unmanned systems, intelligent automation integrated at the design phase will greatly assist in the selection of optimal instrumentation sets. (By optimal instrumentation set we are referring to the set of instrumentation that will provide the most complete information during monitoring, control and fault diagnosis functions).

An area that can be greatly enhanced by intelligent automation is Fault Diagnosis, Fault Isolation and Recovery (FDIR) functions. Intelligent FDIR automation will allow more productive and effective use of crew personnel, fewer system shutdowns, reduced system downtime, improved safety, maintainability, reliability, and reduce crew cognitive overload.

The need for intelligent Fault Diagnosis, Fault Isolation and Recovery (FDIR) automation in several different physical systems has lead to design concepts that are being developed into a methodology and implemented in a generic software tool that will provide automated FDIR functions for any physical system. This generic software tool is the System Diagnostic Engine (SDE). The objective of the SDE project is to define and develop a methodology for design, analysis and implementation of diagnostic systems. The goal of the SDE project is to develop this methodology into a generic, domain independent software tool that will provide automated FDIR capabilities at reasonable execution speeds with database integration and knowledge acquisition capabilities.

The use of a generic software tool like the SDE will augment heuristic-based (rule-based) automation by reducing the brittleness that is inherent in heuristic-based systems. The SDE also has the potential for reducing the initial cost of the system design and automation of the FDIR procedures.

2.0 INTELLIGENT FDIR AUTOMATION

Growing interest in automating FDIR functions is evident in several projects throughout the aerospace community including but not limited to: Faultfinder (1), an intelligent aid for assisting flight crew in FDIR functions; Helix (7), intelligent aide for diagnosing the power train of twin-engine helicopters; and Muxpert (4), intelligent aide for diagnosing AH-64A Apache multiplex subsystems. Each project deals with a different domain but use similar architectures and qualitative reasoning techniques to improve automated FDIR capability.

2.1 Qualitative Reasoning

Qualitative reasoning is an area of Artificial Intelligence (AI) research that addresses problems of reasoning about physical systems. This includes the areas of causal reasoning (9,10,3), reasoning from structural knowledge (9,10,3), qualitative modeling (3), qualitative simulation (8), etc. Qualitative reasoning shows great promise in augmenting the "traditional" expert system technology. (By "traditional" expert system technology we are referring to the technology of developing expert systems using heuristic knowledge). When people reason about physical systems they use more than just experiential (or heuristic) knowledge, they in fact use commonsense knowledge and often develop a mental model (with the appropriate level of detail necessary) to understand the physical system's behavior and to reason about novel faults. Qualitative reasoning research involves automating this human reasoning process. (3,10) A common theme of much of the qualitative reasoning research is explaining how physical systems work using a description of system structure and behavior. The behavioral description of the physical system can be derived from the system structure; structure being the physical system's components, the connectivity between the components, and the component behaviors. (9) The term 'behavior' refers to the observable changes (over time) of the state of the components and the system as a whole. Components have individual behaviors and the collective interactive component behavior results in the behavior of the system as a whole. While the 'structural description' consist of the individual variables that characterize the system and their interactions, the 'behavioral description' consist of the potential behaviors of the system. The 'functional description' of a physical system reveals the purpose of a structural component or connection in producing the behavior of a system. For example, the function of a release valve on a pressurized tank is to prevent an explosion; the behavior of the system as a whole is to maintain a pressure below a certain limit.(8)

Reasoning about the functional description of a physical system can facilitate understanding of the system behavior. This can lead to interesting optimizations in system design and creative alternatives to fault recovery procedures for physical systems. A completely different component may be substituted for a piece of a larger system if the function of the two components are equal. For example, a light bulb could be used to replace (or partially replace) a small heat source. In a fault recovery situation, the location of the replacement component is an additional constraint that must be considered. (The light bulb must be in an appropriate location to be considered for use as a heat source.)

2.1.1 Causal Reasoning

Qualitative reasoning research supports the following reasoning tasks: 1) simulation - starting with a structural description of a physical system, and initial conditions, determine a likely course of future behavior (8); 2) envisionment - starting with a structural description, determine all possible behavioral sequences (6); 3) diagnosis - comparing composed behavior (as computed from a structural description) with specified desired behavior (5); 4) verification - ascertain that a particular implementation structure has a composite behavior which matches the desired behavior specification (2).

A common criteria for explanation in each of these qualitative reasoning tasks involves causal reasoning. Causal reasoning refers to the use of causal knowledge (i.e. cause and effect information) about a physical system to derive knowledge about the behavior and function of the physical system. This reasoning method contrast with standard physics where systems are described by differential equations which provide constraints on the dynamics of the system state variables. Although the analytical techniques are capable of capturing a more complete state of knowledge, people rarely use analytical techniques when reasoning about a physical system. More often people will use causal information in mental models to gain an understanding of system behavior. Since people are very capable of performing FDIR functions without solving differential equations in their heads, it is reasonable to assume that causal reasoning is useful in intelligent FDIR automation. In fact, a great deal of information can be derived from an understanding of component connectivity and causal processes that underly a physical system.

2.1.2 Connectivity Representation

The structural description of a physical system consist of system components, the connectivity between components, and

component behaviors (as discussed in section 2.1). The component connectivity in a physical system structure can be represented using "adjacency" and "reachability". Adjacency describes components that are directly connected to one another (e.g. a resistor is directly connected to a wire). Reachability refers to components that can have an effect on one another but are not adjacent. In figure 1 the regulator is adjacent to the hall-sensor and the hall-sensor is adjacent to the amp-trigger, but there is also a reachable connection (effect) from the regulator to the amp-trigger. There are inherent advantages to "gathering" adjacent and reachable connectivity information about a physical system into a computable structure. "Gathering" information refers to the automatic collection of data from the structural description. This can be accomplished by computing the connected reachability information from the adjacency data which was derived from the structural description.* By "computable structure" we are referring to a stable computer structure (e.g. a matrix) that is developed a single time at the initial execution stage and provides rapid access to data. Using this method allows information associated with data propagating through the components of the system to be readily available without simulating propagation (e.g. tracing through a frame or object type of representation). Examining the reachability data also gives us clues about the physical system's structural description that may not be obvious in a frame or object type representation. (Especially in system representations that contain a large number of interconnected components). An example of this type of undetected representation would be a circuit (as the term applies to graph theory). A circuit can be thought of as a continuous loop in a structure and would have significant impact on data propagating through a physical system. The ability to detect this type of structural information could assist in creating a more complete automated FDIR capability. Having access to the reachability data provides us with a relatively simple method for analyzing diagnostic instrumentation requirements during the physical system design phase. Understanding connectivity in a physical system allows us to better analyze the proficiency of a particular instrumentation set for handling failure coverage in safety critical operations.

Methods for combining the types of reasoning discussed above, and ways to apply qualitative reasoning techniques to permit

*Technical design details are not being published in this paper because of approval difficulties. Details will be included in forthcoming publications.

combinations of qualitative reasoning tasks in automated diagnostic reasoning systems are being studied in the projects listed above (section 2.0) and in the System Diagnostic Engine (SDE) project described in the following paragraphs.

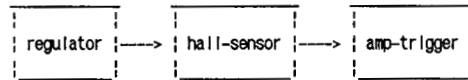


Figure 1 Adjacency/Reachability

3.0 SYSTEM DIAGNOSTIC ENGINE

The idea of a SDE was conceived from attempting to enhance the automated FDIR capability in a heuristic-based expert system by combining causal knowledge and experiential knowledge. (10) The approach to diagnosing a malfunctioning system through the use of a deep understanding of the fundamental structure and behavior of the system and its components has the target of providing an expert's troubleshooting ability without explicitly modeling the expert. The advantage of this approach are especially apparent when automating FDIR operations. Certain aspects of the FDIR operations will change when moving from a manned operational mode to a fully automated operational mode. A human operator may be required to make observations that are unavailable to an automated system. For example, a voltage reading from a meter might be necessary for fault diagnosis in a manned operational mode. An automated system cannot access the same information and must rely on other methods to derive the same results. Integrating intelligent FDIR automation during system design will assist in selecting the optimal set of instrumentation to allow the automated system to have access to appropriate sensor information.

3.1 SDE Architecture

The methodology architecture uses a combination of heuristics, qualitative models, 1st principles and causal information to reason about system structure, functions and associated faults. The methodology will support design of systems for diagnosability and intelligent automated control for FDIR, and will augment heuristic (rule-based) expert system technology by handling cases where rules and procedures are invalidated by unanticipated events.

The knowledge representation architecture (figure 2) will reason about faults in the following manner. The system to be diagnosed is first modeled causally in a hierarchical sense with each level of the hierarchy

showing the connection graph of the component structure. Each component is then modeled as an object which contains an executable qualitative model of its physical behavior. The component object also includes a list of component inputs and outputs, input and output value limits, connectivity information, history (this would include Mean Time Between Failures (MTBF)), and heuristics (rules of thumb) about the component. With the system structure defined to a depth adequate for the system diagnosis (appropriate Line Replaceable Unit (LRU) level), the SDE will start with the root of the hierarchy and determine which subcomponents of the root component are malfunctioning. The SDE will prioritize the list of suspect components using knowledge about connectivity, heuristics, knowledge of system goals, component history and resource limitations. The SDE will then successively call for diagnosis of these components. Finally, the lowest level of the hierarchy will be reached and control would be passed to another portion of the SDE application containing system function knowledge to determine the recovery steps that are necessary.*

Recovery reasoning requires an understanding of the functions a system is required to carry out. This understanding should include the relationship of these functions to the standard goals of the system, to other functions (i.e. functional interdependencies) and to the components that make up the system. A possible abstract depiction of this representation is shown in figure 3. This representation would come into use after the diagnosis had identified the faulty component. The functions that are dependent on faulty components would need to recover either through functionally redundant hardware or a change in the system operations. The exact nature of the functional representation and the reasoning processes that utilize it is future work that must be completed as an extension to the general diagnostic methodology developed today.

4.0 PROTOTYPE DEVELOPMENT

To provide demonstration and proof of concept, the SDE was applied to a small example application; the Manned Maneuvering Unit (MMU) Translational Hand Controller (THC) was chosen for this purpose. (The MMU is the backpack used by the astronauts during Space Shuttle Extravehicular Activities). The following related tasks are currently complete: 1) a minimal core capability of the SDE implemented in KEE on a Symbolics computer, 2) a qualitative simulation of the MMU THC capable of failing one or more components and implemented in KEE, 3) a knowledge base for the MMU THC application (a graphical representation of the structural

description for the MMU THC is shown in figure 4 and the structural description derived from the MMU THC knowledge base by the SDE is shown in figure 5). (11)

Major goals of the SDE implementation include maintaining portability and reasonable execution speeds. If automated diagnostics is to be transferred to real world systems, portability is an important issue. The projects mentioned in section 2.0 and the SDE itself have been prototyped using a variety of powerful software tools and computers. These environments are intended to be rapid prototyping environments and are unlikely to be ported to the final physical system for use in integrating intelligent diagnostics. The SDE has implemented methods, representation structures, algorithms, etc. with the intent to port to hardware and software that can be integrated into physical systems. Although the porting task could require substantial recoding, (e.g. porting to Ada), the underlying design will remain unchanged.

5.0 CONCLUSIONS

A requirement of automated FDIR is the integration of the system design with the intelligent FDIR software. This integration will benefit the system development during all phases of the life cycle by providing 1) the capability to analyze diagnostic instrumentation requirements during the design phase, 2) providing a ready software architecture for implementation of intelligent diagnostics, and 3) providing the opportunity to analyze different instrumentation configurations for failure coverage necessary in safety critical operations. Automating FDIR procedures for a physical system that is already designed can result in difficult problems and unsatisfactory results. Although some level of automation can be obtained for the physical system that is already in use, FDIR operations will probably always require the attention of a human operator (e.g. reading a volt meter as described in section 3.0). If intelligent automation is to be implemented successfully and fully autonomous FDIR capability is desired, it is necessary to integrate intelligent FDIR automation during all phases of the system life cycle. The System Diagnostic Engine (SDE) methodology allows an integrated approach to development of intelligent FDIR.

**ORIGINAL PAGE IS
OF POOR QUALITY**

ACKNOWLEDGEMENT

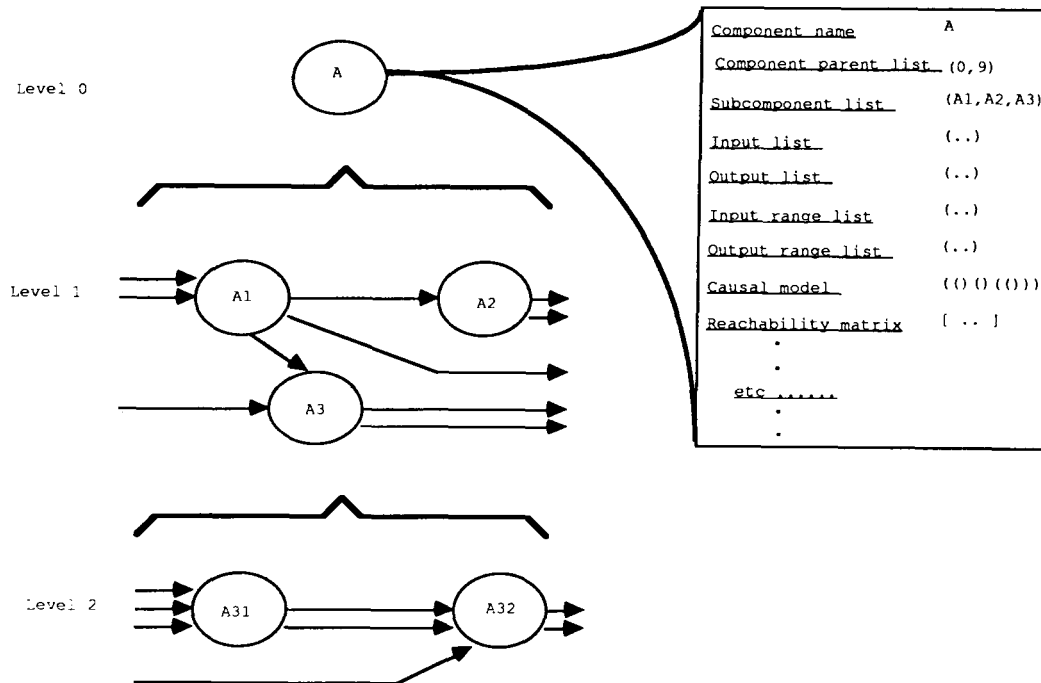
I wish to thank my co-designer who contributed greatly to this work but has requested to remain anonymous for this publication.

REFERENCES

1. Abbott, K., Schutte, P., Palmer, M., Ricks, W., "Faultfinder: A Diagnostic Expert System with Graceful Degradation for Onboard Aircraft Applications", 14th International Symposium on Aircraft Integrated Monitoring Systems, Friedrichshafen, West Germany, September, 1987.
2. Barrow, H., "VERIFY: A Program for Proving Correctness of Digital Hardware Designs", QUALITATIVE REASONING ABOUT PHYSICAL SYSTEMS, MIT Press, Cambridge, Massachusetts, 1986, 437-493.
3. Bobrow, D., "Qualitative Reasoning About Physical Systems: An Introduction", QUALITATIVE REASONING ABOUT PHYSICAL SYSTEMS, MIT Press, Cambridge, Massachusetts, 1986.
4. Bowes, R., Cambell, T., "A Model-Based Approach To MIL-STD-1553 Verification And Diagnosis", American Helicopter Society National Specialists' Meeting on Flight Controls and Avionics, Cherry Hill, New Jersey, October, 1987.
5. Davis, R., "Diagnostic Reasoning Based On Structure And Behavior", Artificial Intelligence, Elsevier Science Publishers B. V. (North-Holland), 24, 1984, 347-410.
6. de Kleer, J., Williams, B., "Diagnosing Multiple Faults", Artificial Intelligence, Elsevier Science Publishers B. V. (North-Holland), 32, 1987, 97-130.
7. Hamilton, T., "HELIX: An Application of Qualitative Physics to Diagnostics in Advanced Helicopters", AAAI Workshop on Qualitative Physics, Urbana, Illinois, May, 1987.
8. Kuipers, B., "The Limits Of Qualitative Simulation", IJCAI 85, Los Angeles, California, August, 1985.
9. Reiter, R., "A Theory Of Diagnosis From First Principles", Artificial Intelligence, Elsevier Science Publishers B. V. (North-Holland), 32, 1987, 57-59.
10. Williams, L., Lawler, D., "Diagnosis: Reasoning From First Principles and Experiential Knowledge", Annual Workshop on Space Operations, Automation and Robotics, NASA/JSC, Houston, TX, August, 1987.
11. Williams, L., Lawler, D., "MMU FDIR Automation Task, Final Report", Contract NAS9-17650, Task Order EC87044, Crew and Thermal Systems Division, NASA/JSC, Houston, TX, February, 1988.

System Structural Representation

Object A Description

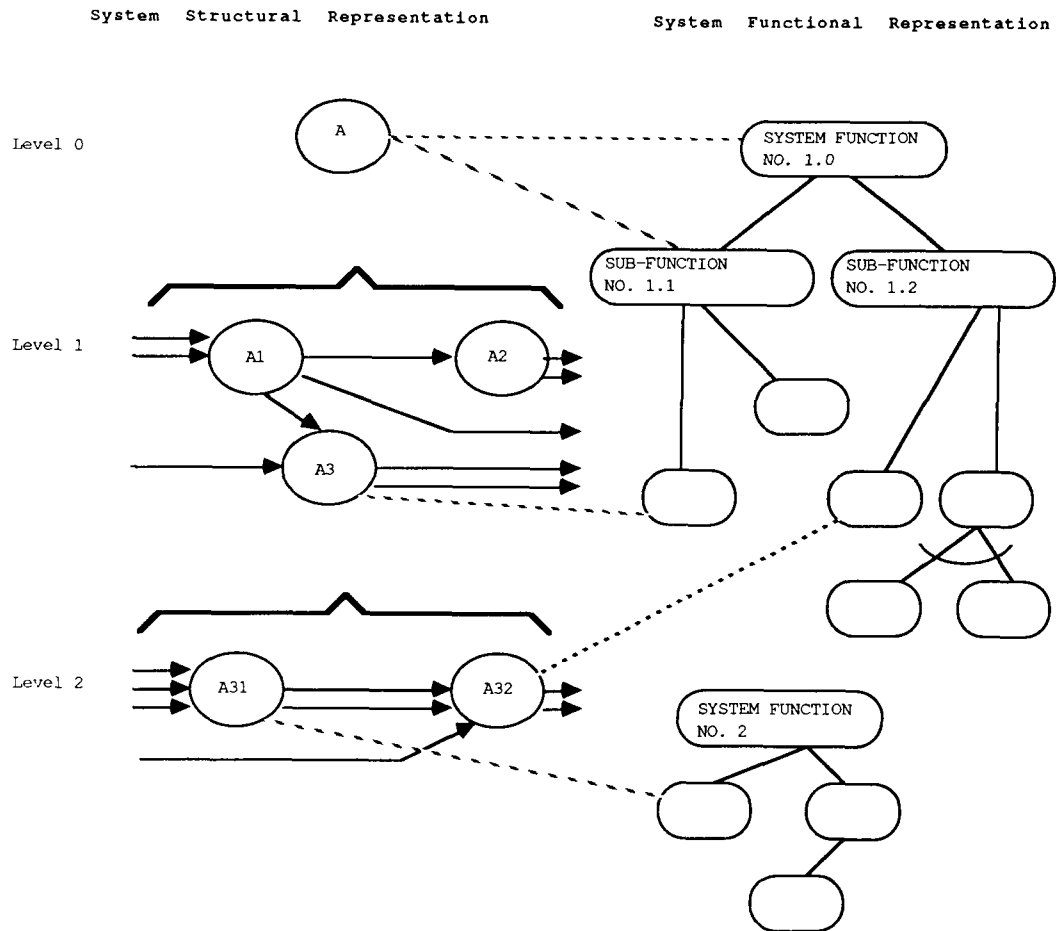


The system is modelled as a hierarchy of directed graphs to whatever depth is needed to provide enough detail for diagnosis. The nodes of the graphs represent the subsystems, assemblies, components, subcomponents, etc. of the overall system. Each node has an object representation (or frame) that contains all information pertinent to the diagnosis.

Figure 2 Knowledge Representation Architecture

ORIGINAL PAGE IS
OF POOR QUALITY

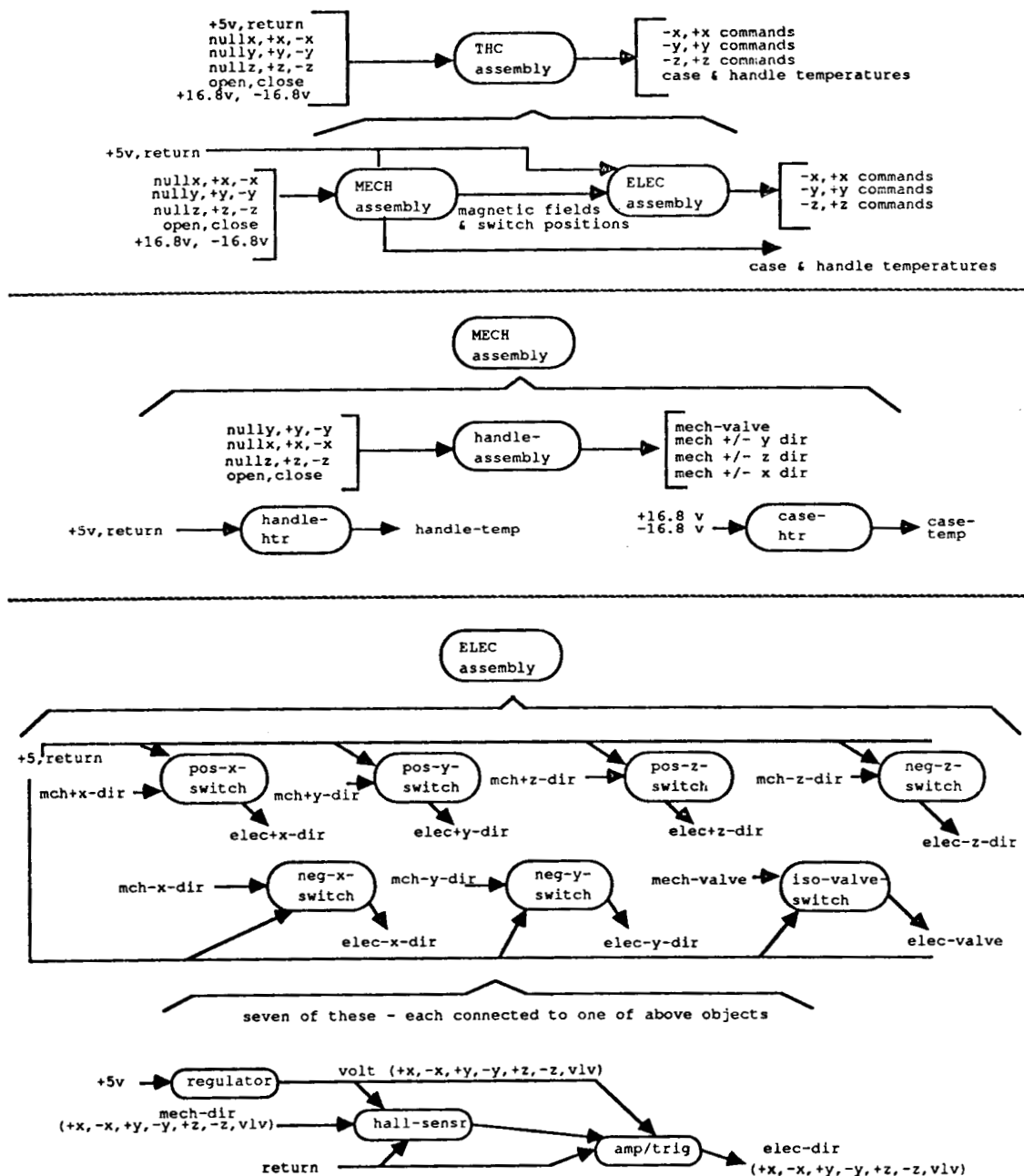
ORIGINAL PAGE IS
OF POOR QUALITY



Possible future extensions of the knowledge representation architecture. The system functionality is specified as an AND/OR graph (not fully-connected). Links are then established between the system structural hierarchy and the functionality graph. These links indicate which system objects are required for the system to be able to carry out particular functions or subfunctions.

Figure 3 Possible Structural/Functional Mapping

ORIGINAL PAGE IS
OF POOR QUALITY



Hierarchical breakdown used to represent THC structure. The top level breaks down into mechanical and electrical assemblies, the 2nd and 3rd level break electrical and mechanical into lowest level physical components.

Figure 4 MMU Hand Controller Structural Architecture

ORIGINAL PAGE IS
OF POOR QUALITY

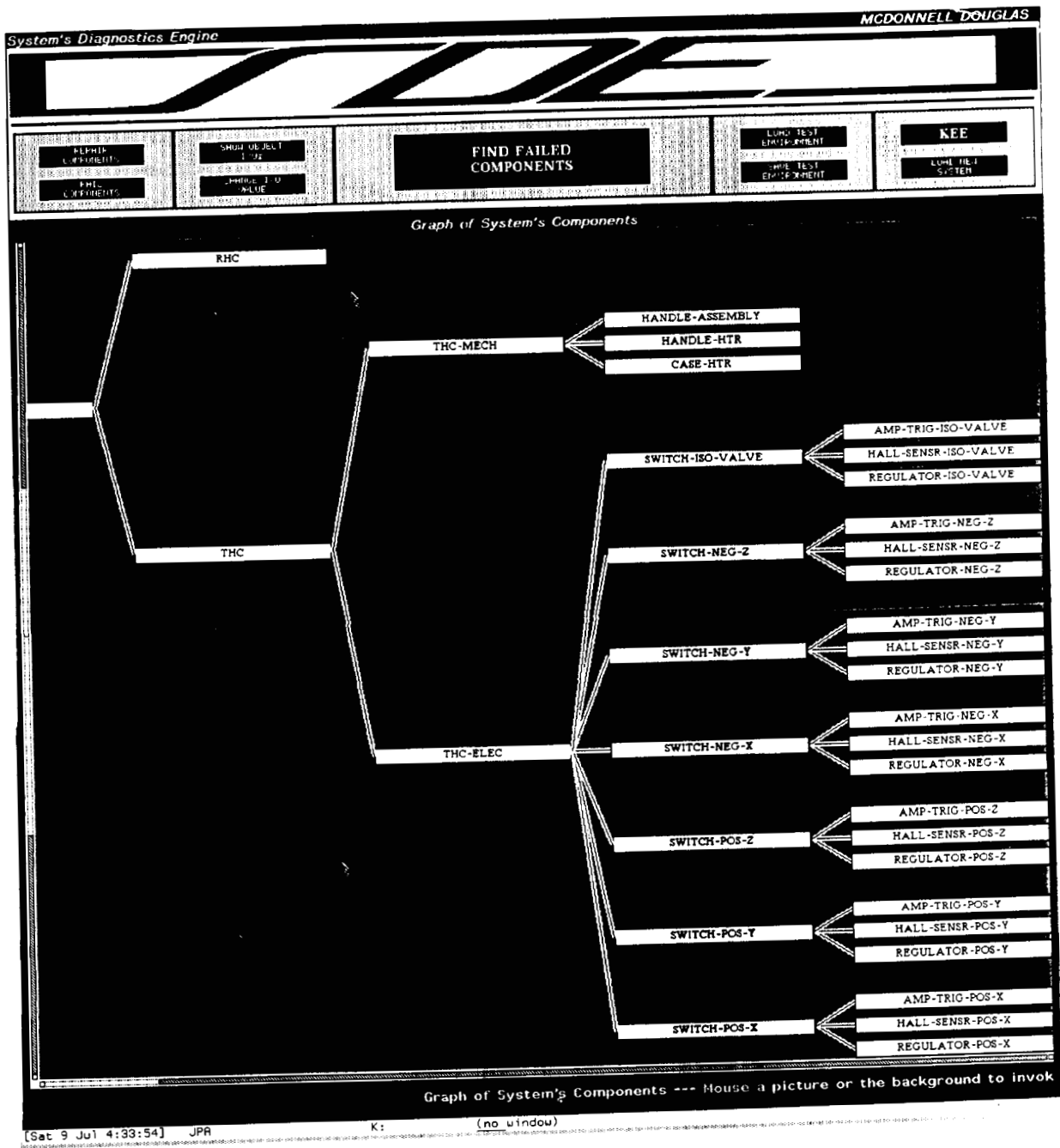


Figure 5 SDE Derived Structural Description
of MMU Hand Controller

**DISTRIBUTED ARTIFICIAL INTELLIGENCE AND
SMART STRUCTURES IN SPACE TRANSPORTATION**

Doug Flaherty
Jacob L. Dickinson
Michael Frank
McDonnell Douglas Astronautics

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

APPLICATION OF FLIGHT SYSTEMS METHODOLOGIES TO THE VALIDATION OF KNOWLEDGE-BASED SYSTEMS

Eugene L. Duke
NASA Ames Research Center
Dryden Flight Research Facility
Edwards, California

ABSTRACT

Flight and mission-critical systems are verified, qualified for flight, and validated using well-known and well-established techniques. These techniques define the validation methodology used for such systems. In order to verify, qualify, and validate knowledge-based systems (KBSs), the methodology used for conventional systems must be addressed, and the applicability and limitations of that methodology to KBSs must be identified. The author presents an outline of how this approach to the validation of KBSs is being developed and used at the Dryden Flight Research Facility of the NASA Ames Research Center.

1 INTRODUCTION

The verification and validation (V&V) of flight-critical systems is a major activity at the Dryden Flight Research Facility of the NASA Ames Research Center (Ames-Dryden). The Ames-Dryden staff assumes safety-of-flight responsibilities for all vehicles flown at the facility. Because these systems are used in research aircraft, the V&V experience at Ames-Dryden is primarily with one-of-a-kind research systems on experimental vehicles. While the range of this experience at Ames-Dryden is somewhat more narrow than that of the validation of flight-critical systems for commercial operations [1], this experience is directly applicable to the types of knowledge-based systems (KBSs) within NASA research programs, whose requirements are to qualify and validate unique, one-of-a-kind research systems.

The Ames-Dryden V&V methodology for embedded flight-critical systems relies on testing,

peer review, abstract models, simulations, and flight validation. This methodology also relies, in large part, on engineering judgment and a tradition that has evolved from the experience with flight-critical systems from the first simplex digital aircraft flight control system on the F-8 digital fly-by-wire (DFBW) aircraft [2], through the triplex DFBW system on the F-8 aircraft [3,4], the 3/8th scale F-15 remotely piloted research vehicle (RPRV) [5], the highly maneuverable aircraft technology (HiMAT) vehicle [6,7,8,9], and the advanced fighter technology integration program AFTI/F-16 [10,11], to the X-29 forward-swept wing aircraft. The result of this evolving, hands-on development of qualification and V&V methodologies is a practical approach that maximizes safety and allows system qualification, verification, and validation to proceed in an expeditious and resource-effective manner.

The V&V methodology used at Ames-Dryden is the same methodology that has actually been used for all flight-critical control systems in non-commercial aeronautical flight vehicles, including the F-18, Space Shuttle, and B-1 aircraft. This methodology uses a subset of the V&V techniques in use or advocated within the aeronautics community. The larger issues of certification and the validation of highly reliable, fault-tolerant systems have been of lesser concern than those of qualifying and conducting flight validation of flight-critical systems.

The basic methodology for the V&V of conventional operation-critical systems is directly applicable to the V&V of KBSs. In fact, if KBSs are to be used in operation-critical applications, the qualification of these KBSs will have to be

PRECEDING PAGE BLANK NOT FILMED

performed within the context of established procedures and will have to address the requirements placed upon the qualification of conventional operation-critical systems. Thus, it is essential that the main features of this well-established V&V methodology be understood.

NOMENCLATURE

a_n	normal acceleration, g
h	altitude, ft
\dot{h}	altitude rate, ft/sec
\ddot{h}	altitude acceleration, ft/sec ²
$f(\ddot{h})$	functional relationship between \ddot{h} and a_n
K_a	aerodynamic gain
K_D	proportional path gain
K_I	integral path gain
δ_{ex}	equivalent longitudinal stick command
Δh	difference between desired and actual altitude
$\frac{1}{s}$	representation of integrator in Laplace variable notation

Abbreviations and Acronyms

AFSR	airworthiness and flight safety review
AFTI	advanced fighter technology integration
AI	artificial intelligence
Ames-Dryden	Dryden Flight Research Facility of the NASA Ames Research Center
CCB	configuration control board
CCR	configuration change request
CDR	critical design review
DFBW	digital fly-by-wire
DR	discrepancy report
FRR	flight readiness review
HiMAT	highly maneuverable aircraft technology
KBS	knowledge-based system
PC	program change (software)
PDR	preliminary design review
RPRV	remotely piloted research vehicle
QA	quality assurance
SDR	system design review

STR	system test report
V&V	verification and validation
WO	work order (hardware)

Definitions

The majority of the following definitions is taken verbatim from Szalai and others [4].

certification The determination by a regulatory authority that a product meets the regulations for that product.

embedded system A system that is an integral part of some larger system. This distinction is particularly important when a subsystem interacts with the larger system in such a way that a failure in the embedded system can propagate to the larger system or cause the larger system to fail.

fault tolerant A system which is able to continue to provide critical functions after the occurrence of a fault.

flight critical A component or system whose failure could cause loss of the aircraft.

mission critical A component or system whose failure could result in the inability to perform a mission.

operation critical A component or system whose failure could result in loss of the aircraft, loss of life or limb, compromise public safety, result in substantial financial loss, or inability to perform a mission.

qualification A formal process whereby a system or aircraft is defined to be ready for flight operations.

system An entity of fixed identity united by some form of purpose, interaction, or interdependence that can be meaningfully isolated.

validation The determination that a resulting product meets the objectives that led to the specification for the product. This determination usually includes operation in a real environment.

verification The determination that a design meets the specification. Verification is usually a part of the validation process. A simulated environment is often used.

2 A METHODOLOGY FOR CONVENTIONAL, EMBEDDED FLIGHT-CRITICAL CONTROL SYSTEMS

The basis of the Ames-Dryden flight qualification and V&V methodology for embedded flight-critical systems is the incremental verification of system components, integration testing, configuration management, and flight validation. The application of the verification, integration testing, and flight validation is discussed within the overall context of the system life cycle. The configuration management aspect of the qualification and V&V methodology is discussed separately.

2.1 Verification, Validation, and the System Life Cycle

Verification and validation is an ongoing process that is an integral part of the system life cycle. The system life cycle for conventional flight control systems is often characterized as a series of stages (figure 1).

When functional specifications are derived from the system goals and requirements, those specifications must be critically examined to establish that the specifications adequately address the system goals and requirements. Similarly, the design specifications must meet the functional specifications. This critical examination is accom-

plished by system design reviews (SDRs), preliminary design reviews (PDRs), and critical design reviews (CDRs). The SDR is a presentation and review of the conceptual design of the system; the goals and requirements are addressed and top-level definition of functional specifications are provided. The purpose of the SDR is to ensure that the understanding of the goals and requirements for the system is consistent between the requesters and designers. The PDR is a presentation of a first-order definition of the system design including a presentation of how the functional specifications are being addressed in the design. The CDR is a detailed design review in which a functional design is presented for review. (Theoretically, no hardware or software is to be implemented until after the CDR, but in practice, system components are implemented early in the life cycle, often before the SDR, to test ideas and may be directly incorporated or modified for the system as finally implemented.) At each of these reviews, designs are presented to a large audience with broad interests ensuring that system-level goals and requirements are addressed, user requirements are satisfied, and safety issues are adequately considered. The review boards provide detailed feedback to the system designers and implementers, and weaknesses or criticisms raised at a design review must be addressed at the next level of review.

The design review process is an iterative, and, one hopes, convergent process in which the goals

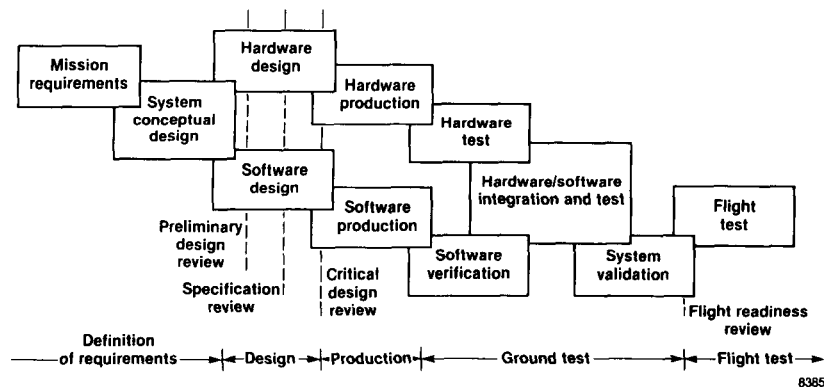


Figure 1. Ames-Dryden Life Cycle for Research Systems

and requirements and functional specifications are interpreted and translated into a design specification that, among other things, establishes a partitioning of functions between hardware and software. Design specifications for both hardware and software are translations of the functional specifications which in turn embody the system-level goals and requirements. The design specification is supported by an interface definition establishing both the interfaces between the system and its environment and the interfaces between the hardware and software.

The design specifications are transformed into hardware and software realizations. This transformation is not a straightforward, one-step process. The transformation of a design specification to an implemented prototype system requires the development and testing of numerous software procedures and hardware circuits, each of which is a prototype of some element in the larger system.

The implementation of system elements or components is supported by a variety of analysis tools and testing techniques [1, 2, 4, 12, 13]. The analysis tools used include failure modes and effects analysis, independent review, static verification, independent calculations, conjectures, and suspicions. This analysis is conducted on abstract models of the system or of the system components. Linear system models, aggregate system models, block diagrams, flow diagrams, schematics, source programs, specifications, and simulations are some of the main abstract models used. This analysis of abstract models is used to translate requirement and design specifications into a physical realization.

The physical realization of a system is constructed from physical realizations of system components such as circuits, microprocessors, computers, and software modules. Hardware components are often breadboard, brassboard, or nonflight-qualified versions of the actual flight system; these hardware components are bench tested in isolation and then incorporated into "hot-bench" test facilities that incorporate other simulated or actual physical systems. Typical of these hot-bench test facilities are simulations incorporating flight hardware (hardware-in-the-loop simulations) [6, 7, 8] or iron bird facilities based on extensive replication

of flight systems and are often based on the use of decommissioned aircraft [4].

Simulation testing provides a closed-loop facility wherein the system is exposed to an environment that closely resembles the electronic and data environment in which the system must actually operate. Simulation also provides a facility for testing that the hardware and software of the system are integrated and operating together. The realism of the simulation is determined by the operating requirements for the flight application of the system (see section 2.3). Simulation is where the pilot (the system user) is first exposed to and allowed to evaluate the system.

This analysis, testing, and verification along with the configuration management process (see section 2.2) constitute main components of the qualification process wherein a system is determined to be ready for flight test. These results are presented to a flight readiness review (FRR) team composed of nonproject engineers from multiple engineering disciplines who perform an independent and in-depth review of the system design, analysis, test results, and configuration management. The FRR team is empowered to recommend additional analysis, testing, or documentation. The results of the FRR are presented at an airworthiness and flight safety review (AFSR) panel where the project team seeking authorization for approval to begin flight testing responds to the findings of the FRR. The AFSR panel is typically composed of engineering and operations managers and flight safety personnel. Only after the AFSR is satisfied is a system taken to flight.

Flight validation is an extension of the testing methods performed on physical models in the simulation. For a research system, the flight tested component is a physical model of itself; if the system is to be fielded in an operational environment, the flight tested system is often a prototype of the final system. During flight test, the system is exposed to the total physical, electronic, and data environment in which it is designed to operate.

Gault and others [1], Holt and others [12], and Hopkins [13] propose the use of additional abstract models (aggregate models) and analysis methods (formal proofs and statistical analysis). These models and analysis tools address one of the chief

limitations in the Ames-Dryden methodology: the reliance on testing, both failure modes and effects and nominal condition testing. This becomes a serious concern when considering either highly reliable, fault-tolerant systems or highly complex systems. Hartmann and others [13] describe the number of tests required for such systems as a *fundamental* problem:

The fundamental problem of fault tolerance validation is the vast number of test cases when all possible combinations of flight conditions and multiple faults are considered.

This view is confirmed by Gerhart and others [1], Holt and others [12], and Gerhart [15], who claim that exhaustive testing is not possible for any but the simplest of systems.

2.2 Configuration Management

Configuration management is the orderly and systematic process of ensuring consistency in development, documentation, testing, problem reporting, and maintenance of a system. The use of a configuration control board (CCB) with review and change approval authority, consisting of representatives from several engineering disciplines, is a key feature of configuration management. Petersen and Flores [16] describe the configuration management process:

The primary purpose of the software control and system configuration management process for flight-critical digital flight control systems is to provide a method for efficient flight system development and a procedure for assuring safe flight operations. The process is designed to control system configuration changes by managing the primary system development phases ... and to resolve discrepancies uncovered during system testing. In addition, the configuration control process prescribes stringent test and documentation requirements and provides for visibility of changes across all involved engineering disciplines through formal review procedures.

Petersen and Flores [16] also present block diagrams showing the steps in the software control and system configuration process (figure 2) and the documentation flow and tracking process (figure 3) used at Ames-Dryden. This process is initiated when the system is put under configuration control which is generally well into the system development cycle.

Figure 2 shows how new system requirements or anomalous system behavior are accommodated in the software control and system configuration management process. New system requirements are introduced into the configuration management process using a configuration change request (CCR); anomalous system behavior is recorded on a discrepancy report (DR) (figure 3).

The use of DRs to record any anomalous behavior is useful for identifying and correcting problems that result from operator error, initialization, or system design. Additionally, the extensive use of DRs provides a means of isolating incipient problems by identifying areas or functions in the system that are repeatedly involved in or associated with anomalous behavior. Tracking DRs also facilitates one aspect of the process of building confidence in the system (see section 2.1): it provides a means of judging the maturity of a system through experience with that system over an extended period. Typical experience with systems as a function of time is shown in figures 4 and 5.

It is important to note that the problems identified in every DR cannot be or are not always remedied. A problem that occurs in a test facility might be highly unlikely in flight; the subject of a DR might even be based on a rethinking of the system design that uncovers a failure mode or potential problem. Program schedule slippage or the costs of fixing the problem are often overriding concerns. The effect of problems identified in DRs is evaluated in terms of the risk associated with them. Those known problems that are identified on DRs and not remedied are called "accepted risks." Accepted risks are always clearly identified before flight testing. The risks associated with these problems are made visible to, and are evaluated by, independent reviewers such as those comprising the AFSR panel (see section 2.1).

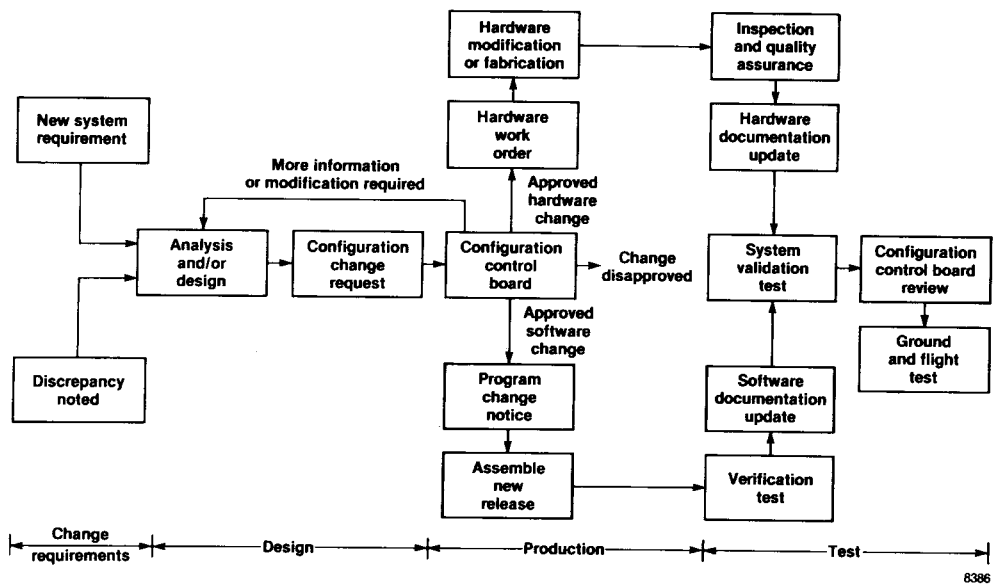


Figure 2. Software Control and System Configuration Management Process

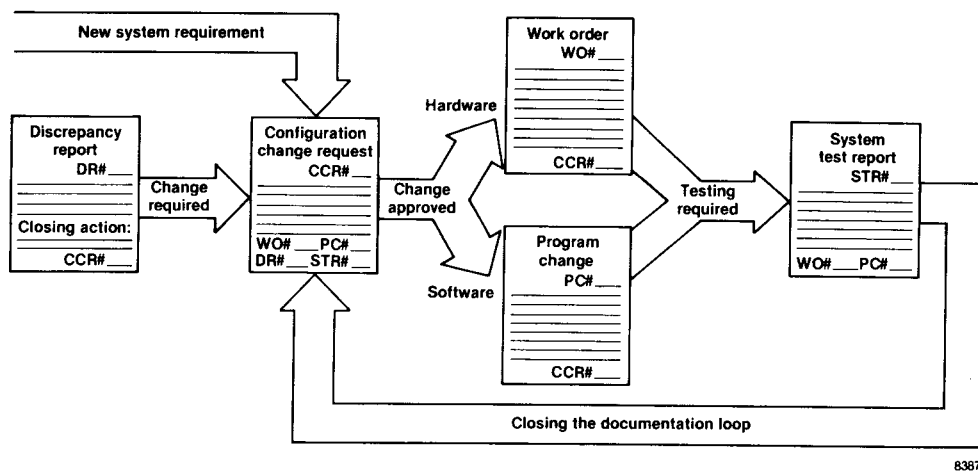


Figure 3. Documentation Flow and Tracking Process

ORIGINAL PAGE IS
OF POOR QUALITY

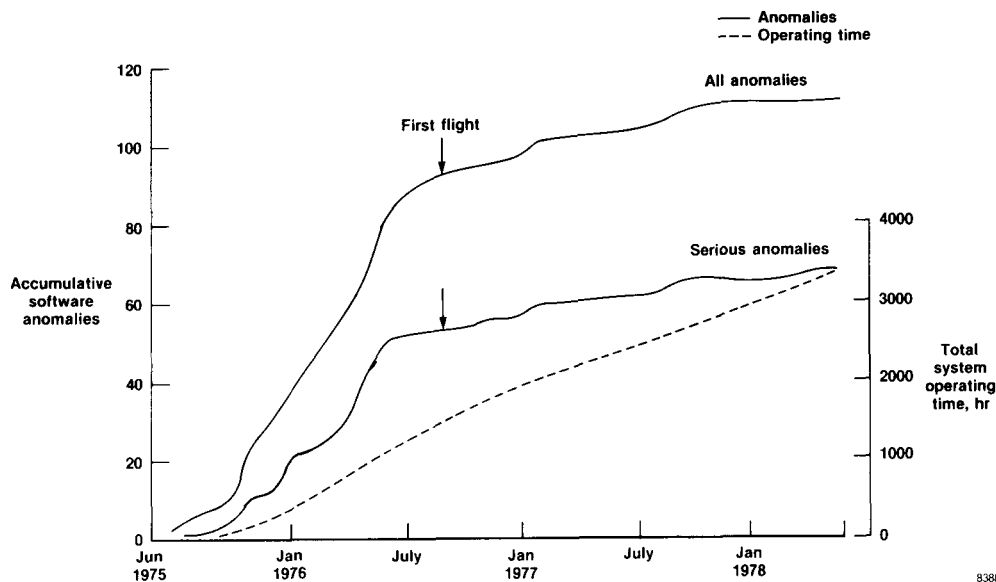


Figure 4. F-8 DFBW Software-Anomaly Experience

2.3 System Criticality and Its Impact on Validation

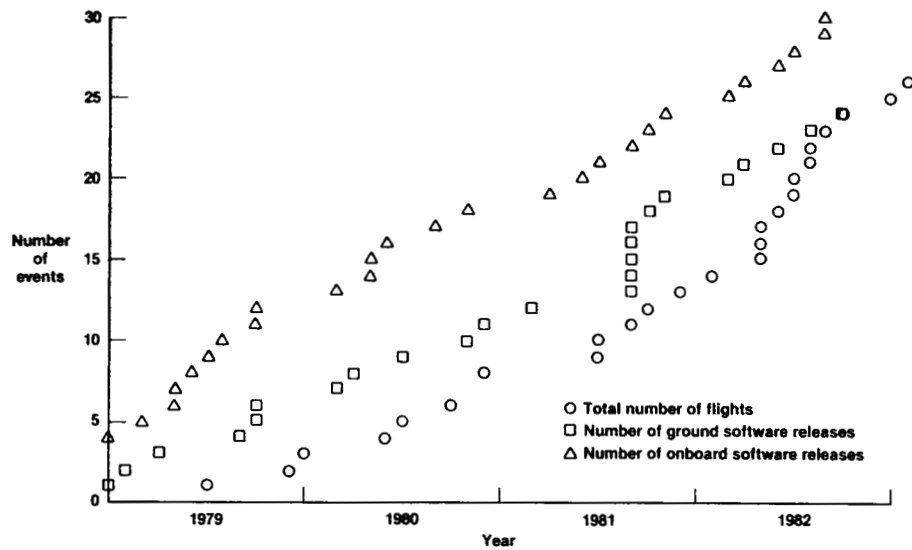
The requirements imposed on the V&V and configuration management process are determined by the criticality of the system. For aircraft flight systems, three levels of criticality are generally recognized:

- A. Systems whose failure could cause loss of life or limb, compromise public safety, or result in substantial financial loss;
- B. Systems whose failure could cause mission failure (mission-critical);
- C. Systems whose failure could cause inaccurate results or inefficient use of resources.

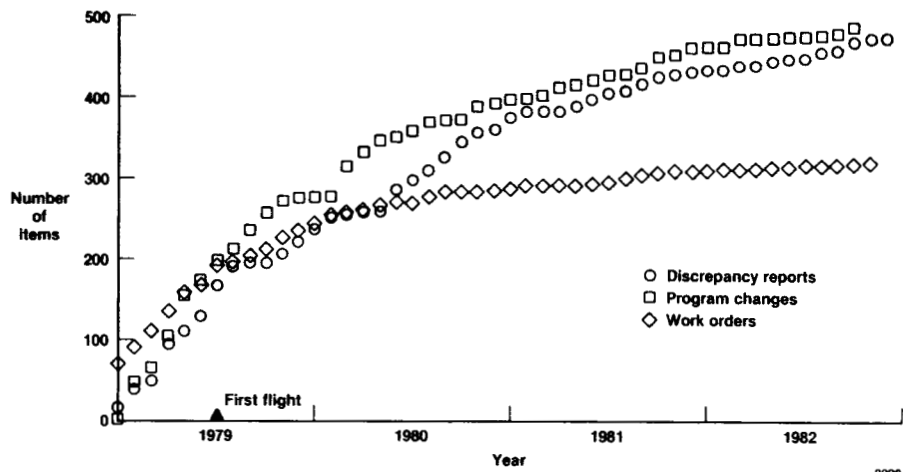
The level of criticality of a system is primarily determined by those requesting that the system be developed. However, system designers or any of the independent review teams can modify that determination. In practice, we have found that it is

usually easier and less work to classify a system as a level B rather than try to support and defend a classification at level C. Nontechnical factors are often taken into account when determining at which level to class a system. A system might be treated at level A when that system or the project of which it is a part is highly visible and any perceived problem might jeopardize research goals.

The configuration management process increases in formality at each higher level of criticality: the composition of the CCB is broader and consists of more members, the requirements for documentation increase, and testing requirements for system changes are more extensive. The requirements for the simulation also increase with higher levels of criticality: a level C system might be qualified off-line, using interactive or batch simulations and stand-alone hardware tests; a level B system requires at least a real-time, piloted simulation for closed-loop testing; and a level A system generally requires a hardware-in-the-loop simulation or an iron bird.



(a) Flights and software releases.



(b) Discrepancy reports, program changes, and work orders.

Figure 5. HiMAT System Development History

3 APPLICATION OF VALIDATION METHODOLOGY TO KNOWLEDGE-BASED SYSTEMS

The V&V methodology used for conventional, embedded operation-critical flight systems provides an established and accepted set of procedures upon which a methodology for KBSs can be based. While this position may be controversial in the AI community, the political and sociological realities of flight research and testing will ultimately dictate that any methodology for the validation of KBSs at least address the currently used methodology for conventional systems.

3.1 The Life Cycle Model for Knowledge-Based Systems

The proposed approach to the V&V of KBSs relies on the life cycle model shown in figure 1. The life cycle model for a KBS has been a topic of considerable concern to some who have addressed the validation of a KBS, and several models have been proposed [17, 18, 19]. These models stress the development and prototyping process in a KBS. The motivation for developing these models is apparently to address the lack of a clear or well-defined statement of system goals and requirements and to highlight the prototyping process common in the development of KBSs. While the proponents of these models would probably contend that there is a fundamental difference between the life cycle of a KBS and a conventional system, another view is that this apparent difference is more reflective of the maturity of KBSs rather than of anything fundamental.

Because KBSs are just emerging in operation-critical applications, there is little certainty of capabilities and limitations of these systems. The prototyping that is a common feature in the development of a KBS often represents an attempt to establish requirements for a given application. This definition of requirements, capabilities, and limitations through prototyping is not unlike that used in conventional systems when new techniques or applications are attempted. The difference is

in the body of knowledge and experience behind the use of conventional systems as opposed to that for KBSs. Also reflected in this prototyping is the lack of maturity of artificial intelligence (AI) techniques in general that provides little basis for the selection of control and knowledge representation methods.

3.2 Problems in the Verification and Validation of Knowledge-Based Systems

There are several issues that are almost certain to create problems for anyone attempting to validate operation-critical KBSs. Perhaps the most serious of these is an unwillingness to treat the current generation of KBSs out of the context of the promises of AI. The current generation of KBSs are not, in general, capable of learning or even modestly adaptive. These systems exhibit few nondeterministic properties. These KBSs may be complex but they are not unpredictable. But so long as there is this persistence in dwelling on the ultimate potential of AI systems instead of on the realities of the system being qualified, it is unlikely that an AFSR panel would allow flight testing.

A further difficulty arises from the contention that KBSs do not always produce the correct answer. If this is true then a KBS can only be used for tasks in which their performance can be monitored and overridden by a human. Most operation-critical systems are required to perform without human intervention or with only high-level supervision or control. However, a KBS that does not always produce the optimum answer is acceptable as long as it never produces a wrong answer. This latter point is in fact one of the main V&V issues: operation-critical systems must be shown to always produce acceptable solutions.

3.3 A Proposed Approach to De- velop a Verification and Valida- tion Methodology for Knowledge- Based Systems

In order to validate a system, one must have a set of requirements for that system, and those requirements must establish the performance criteria and the limitations of the system. The cur-

rent claim from some within the AI community that many of the characteristics of AI systems preclude such requirements either do not understand the validation issue or are unwilling to accept the structure and formalism required for validation. To address the issue of requirements, an incremental approach to validating KBSs is needed.

There are two key aspects of the proposed approach to the V&V of KBSs:

1. development of a KBS to perform some task that is well-known, well-understood, and for which conventional V&V techniques are adequate; and
2. incrementally and simultaneously expand both the KBS and the V&V techniques to more demanding and complex tasks.

The procedures used for verifying, qualifying, and validating conventional operation-critical flight systems at Ames-Dryden will be applied and modified as required. Because we ultimately plan to carry these experiments to flight using the rapid-prototyping facility [20], this process will be performed under the aegis of the AFSR panel and will be under periodic review. The subject of this research will be a KBS that is being developed to perform aircraft maneuvers normally performed by highly trained pilots.

The research plan is to identify maneuvers of increasing difficulty and to build gradually more complex and adaptive KBS to accomplish those maneuvers. This will include prototyping, evaluation, and a series of initial operating capabilities

that will evolve into a sequence of documented requirements for testing against each version of the system. This approach fits well within the model of and practice used with conventional digital systems.

4 VALIDATING A SIMPLE KNOWLEDGE-BASED SYSTEM

To illustrate the proposed approach to the V&V of KBSs, a rule-based longitudinal altitude-command autopilot example for an F-15 aircraft will be presented. The example presented represents a single axis of a three-axis (longitudinal, lateral-directional, and velocity axes) controller. This controller is being developed and will be qualified as a mission-critical system (see section 2.3) as part of the research into validation methodologies for operation-critical KBSs.

4.1 Goals and Requirements for Example Knowledge-Based System

A simplified representation of the aircraft and control system is shown in figure 6. The objective is to develop and to demonstrate a knowledge-based controller that produces command inputs to the aircraft control system based on a dynamic world model obtained from instruments on the aircraft and on a simple set of rules. While this task

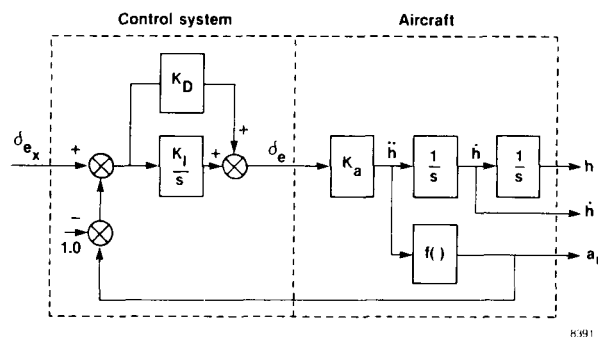


Figure 6. Simplified Longitudinal Model of the F-15 Aircraft and Its Control System

may not represent a suitable application of a KBS (because it is easily performed by conventional algorithmic control laws), it provides a simple mission-critical application that is both easy to understand and easy to validate.

The control task requires the autopilot system (whether based on conventional algorithms or a knowledge-based approach) to produce commands that cause the measured aircraft altitude h to be within some specified tolerance Δh of the commanded altitude h_{com} . Additionally, constraints are placed on the altitude rate \dot{h} and the normal acceleration a_n . The constraint on a_n is the same as a constraint on altitude acceleration \ddot{h} , but a_n represents a more easily understood and easily measured physical quantity.

The initial requirement for this controller was that it control the aircraft in a consistent, repeatable manner at least as well as a pilot during both the transition mode (going from one altitude to another) and the altitude-hold mode (controlling the aircraft about a specified altitude). The desire was to have it control the aircraft as well as a conventional algorithmic autopilot. An additional goal was to allow off-condition engagement so that the controller would be effective even without benign initial (engagement) conditions.

These goals and requirements are similar to those initially imposed on the altitude-hold capabilities of the flight test maneuver autopilot for the HiMAT vehicle [21]. The constraints and tolerances were established as baseline figures. From this initial specification, a rule-based system was implemented that combined numeric and symbolic methods. This initial system was tested using a detailed nonlinear simulation model of the aircraft and its control system; the controller achieved excellent results for some initial conditions but performed poorly for many others. This initial result was typical of that experienced when evaluating the initial implementation of a conventional controller on a nonlinear simulation. After several iterations of this process, a fairly detailed statement of performance capabilities and limitations was established (table I). This information, in essence, represents a clarification of the statement of goals and requirements, serves as the basis of a functional specification for the system, and defines the

system test matrix.

4.2 Life Cycle of Example Knowledge-Based System

By this point in the life cycle, the development of a conventional controller would be supported by design and analysis tools and abstract (linear) models of not only the aircraft and its control system but of the controller as well. These tools and models would provide some of the basis of the validation of a conventional system by establishing metrics of system performance and robustness. The main benefit of having such tools and models is that their use allows extensive testing with a minimum of computational expense; only selected test points need to be repeated using the nonlinear simulation. For the rule-based controller, tools and analysis techniques either do not exist or are rudimentary at best. This difference in development will create some difficulties in qualifying the system for flight. Parts of the problem are both technical and sociological. Verification will have to rely on more extensive testing and a thorough exposition of the nature of the rules. The testing will require that a large number of tests be conducted on the nonlinear simulation that extends the time required for conducting those tests.

Table I. System
Performance Capabilities
and Engagement Conditions
Defined by Prototyping

Performance requirements	
Δh	$= \pm 50 \text{ ft}$
\dot{h}_{max}	$= \pm 100 \text{ ft/sec}$
$a_{n_{pos}}$	$= 2.0 \text{ g}$
$a_{n_{neg}}$	$= 0.5 \text{ g}$
Engagement conditions	
Δh	$= \pm \infty \text{ ft}$
\dot{h}_{max}	$= \pm 200 \text{ ft/sec}$
a_n	$= \pm 2.0 \text{ g}$

The next step in the life cycle is an SDR. This has been conducted informally during development but now requires formal exposure and review. The rules derived from prototyping (table II) and a detailed definition of the verification

test matrix will be presented and reviewed at the SDR. Again, this addresses both the technical and sociological aspects of V&V: the SDR provides a technical assessment of the design, allowing the completeness and consistency of the rules to be examined by independent reviewers and serves as a gentle introduction to the idea of using KBSs in such applications.

Table II. Rules for Longitudinal Altitude-Hold Autopilot

Performance boundary rules*
<ul style="list-style-type: none"> • If the altitude acceleration exceeds the positive acceleration limit, move stick forward. • If the altitude acceleration exceeds the negative acceleration limit, move stick aft. • If the predicted altitude rate exceeds the positive altitude rate limit, trim stick forward. • If the predicted altitude rate exceeds the negative altitude rate limit, trim stick aft.
Normal command rules*
<ul style="list-style-type: none"> • If the altitude error is positive and the predicted altitude rate is negative, trim stick aft. • If the altitude error is negative and the predicted altitude rate is positive, trim stick forward. • If the predicted altitude error is positive and the altitude error is small, click stick forward. • If the predicted altitude error is negative and the altitude error is small, click stick aft. • If the predicted altitude error is positive and the altitude error is large, trim stick forward. • If the predicted altitude error is negative and the altitude error is large, trim stick aft.

*Definitions:

move large movement of stick
trim intermediate movement of stick
click small movement of stick

It is expected that the development of this rule-based controller will continue through the

normal life cycle for research systems. The main differences that are expected between conventional and KBSs are that for the KBS

1. the design reviews will serve both educational and technical purposes;
2. the design will incorporate more problem specific experience (but probably less fundamental system understanding) at each stage in the life cycle;
3. the lack of traditional tools and abstract models will force earlier recognition and definition of system testing requirements; and
4. because of the lack of tools and abstract models, the testing required for the rule-based system will be more extensive than that required for a conventional system of similar capabilities.

4.3 Test Matrix for Example Knowledge-Based System

To appreciate the number of individual tests that must be performed as part of the validation of this longitudinal autopilot, two factors must be understood:

1. the performance and limitations define a matrix of test conditions for each simulated flight condition; and,
2. because the dynamics of an aircraft vary throughout its flight envelope, that matrix of test points must be repeated at many flight conditions.

The performance requirements and engagement conditions define the requirements for both on- and off-condition operation. To test the on-condition requirements for the example autopilot, one engages the system at the test altitude and Mach number and monitors the performance of the system to ensure that none of the performance limits are exceeded. The testing of engagement requirements requires a set of tests about each of the altitude and Mach number points. Thus, for a given altitude and Mach number, the system must be engaged at a number of conditions representing

the permutations of the bounds of the engagement conditions; again, time histories are monitored to ensure that the system performs within the limits established by the performance requirements. At each altitude and Mach number test condition, this requires a minimum of eight separate tests.

The dynamics of an aircraft are not constant throughout the flight envelope. To ensure that the system performance goals are met, tests must be performed at a number of flight conditions (figure 7). At each altitude and Mach number condition, the entire matrix of performance requirements must be tested at the engagement limits.

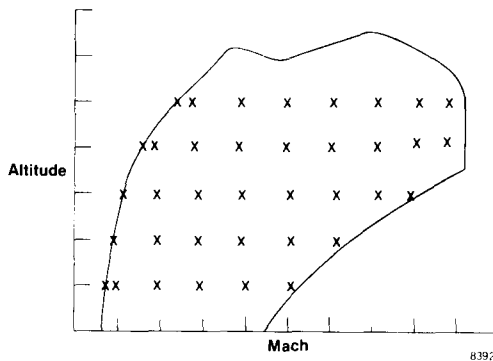


Figure 7. Typical Flight Envelope With Example Test Conditions

This testing is time consuming and requires a detailed nonlinear simulation. A conventional system would require less simulation testing on the nonlinear simulation because it would be supported by abstract models of the aircraft and the autopilot. The nonlinear simulation would be used at a few selected altitude and Mach number conditions to verify the abstract models.

It is important to note that the testing described above

1. includes no failure condition testing,
2. the example autopilot is a greatly simplified representation of a system that will be taken to flight, and
3. the rules presented in table II represent only a single axis of a three-axis controller.

5 LIMITATIONS OF VALIDATION METHODOLOGY FOR KNOWLEDGE-BASED SYSTEMS

The most serious limitation of applying the V&V methodology for conventional systems to operation-critical KBSs is the lack of both structured development methods and verification tools and techniques. Conventional systems are supported by design and analysis tools and techniques, coding standards, and methods for examining software that is procedural in nature. These tools, standards, and procedures do not exist for KBSs nor are any likely to emerge in the near term. Another limitation of applying the conventional V&V methodology to KBSs is that component testing is difficult if not impossible. Both of these limitations will force validation to rely on integrated system testing, treating the total KBS as a black box.

The testing requirements for a system do not increase linearly with the complexity of the system; testing requirements grow as a polynomial or exponential function of system complexity. As a simple example of the growth of the test matrix with system complexity, consider the test matrix defined for the longitudinal autopilot (see section 4.3). A similar matrix would be defined for each axis of the total autopilot. If we assume that there are m tests required for each axis, then the final autopilot will have three axes of comparable complexity; the total number of tests will be m^3 because all combinations of tests will have to be performed at each flight condition to validate the system performance.

Testing any but the most simple systems as black boxes requires a test matrix of overwhelming complexity. This will compound an already severe problem that has been a consistent factor in the V&V of conventional systems: the cost, schedule, and personnel requirements for V&V greatly exceed the development costs and almost always cause programmatic delays. Further, the costs and delays are directly related to how late, in the de-

velopment cycle, design and implementation errors are detected.

One of the main challenges of developing a validation methodology for KBSs is to develop tools and techniques that will allow highly complex systems to be verified, qualified for flight, and validated in a cost-effective and timely manner without having to reduce the capability or operational envelope of that system. (This challenge, incidentally, is one that those working with conventional systems must also face.) As part of the Ames-Dryden effort in developing and demonstrating a viable validation methodology for KBSs, the development of automatic testing systems is an integral part. The goal of this effort is to generate test matrices automatically from requirements and specifications for use in an automated testing system capable of both conducting tests and monitoring and interpreting test results.

6 CONCLUDING REMARKS

The qualification, verification, and validation methodology used at Ames-Dryden for flight-critical control systems and how this methodology can be extended and applied to intelligent knowledge-based systems are reviewed in this paper. The justification for the use of this methodology is the similarity of the current generation of KBSs with conventional systems in terms of complexity and function. Limitations of the proposed methodology for both highly reliable, fault-tolerant systems and extremely complex systems such as might be envisioned for future generations of KBSs are discussed. Research and development areas are suggested to augment and enhance the current methodology to support both conventional systems as well as KBSs.

The main differences between conventional systems and KBSs are that for the latter

1. the design reviews will serve both educational and technical purposes,
2. the design will incorporate more problem-specific experience (but probably less fundamental system understanding) at each stage in the life cycle,

3. the lack of traditional tools and abstract models will force earlier recognition and definition of system testing requirements, and
4. because of the lack of tools and abstract models, the testing required for the rule-based system will be more extensive than that required for a conventional system of similar capabilities.

The view presented in this paper is consistent with that proposed in Gault and others [1]:

A validation methodology for such systems [ultrahigh reliability, fault-tolerant systems] must be based on *a judicious combination of logical proofs, analytical modeling, and experimental testing.*

This methodology must be supported by reliable, validated development and test tools that lower the cost and reduce the schedule, if the goal of validation is to be achieved for either highly reliable, fault-tolerant systems or highly complex systems such as are envisioned for KBSs.

Perhaps the biggest obstacle in the qualification of operation-critical KBSs is the mystification and obfuscation by the advocates and developers of KBSs. While stressing the enormous differences between KBSs and conventional systems may be a useful tactic in generating enthusiasm and support for the development and use of KBSs, this approach is almost guaranteed to discourage acceptance and prevent deployment of these systems in operation-critical applications.

REFERENCES

1. Gault, J.W., Trivedi, K.S., and Clary, J.B., Eds., "Validation Methods Research for Fault-Tolerant Avionics and Control Systems — Working Group Meeting II," NASA CP-2130, 1980.
2. "Description and Flight Test Results of the NASA F-8 Digital Fly-By-Wire Control System — A Collection of papers from the NASA Symposium on Advanced Control Technology, Los Angeles, California, July 9-11, 1974," NASA TN D-7843, 1975.

3. Szalai, K.J., Felleman, P.G., Gera, Joseph, and Glover, R.D., "Design and Test Experience with a Triply Redundant Digital Fly-By-Wire Control System," AIAA-77-1042, 1977.
4. Szalai, K.J., Jarvis, C.R., Krier, G.E., Megna, V.A., Brock, L.D., and O'Donnell, R.N., "Digital Fly-By-Wire Flight Control Validation Experience," NASA TM-72860, 1978.
5. Edwards, J.W., and Deets, D.A., "Development of a Remote Digital Augmentation System and Application to a Remotely Piloted Research Vehicle," NASA TN D-7941, 1975.
6. Evans, M.B., and Schilling, L.J., "The Role of Simulation in the Development and Flight Test of the HiMAT Vehicle," NASA TM-84912, 1984.
7. Myers, A.F., and Sheets, S.G., "Qualification of HiMAT Flight Systems," 7th Annual Technical Symposium, Association for Unmanned Vehicle Systems Proceedings, Dayton, Ohio, June 1980.
8. Myers, A.F., Earls, M.R., and Callizo, L.A., "HiMAT Onboard Flight Computer System Architecture and Qualification," Journal of Guidance, Control, and Dynamics, Vol. 6, No. 4, 1983, pp. 231-238.
9. Petersen, K.L., "Flight Control Systems Development of Highly Maneuverable Aircraft Technology (HiMAT) Vehicle," AIAA-79-1789, Aug. 1979.
10. Mackall, D.A., Ishmael, S.D., and Regenie, V.A., "Qualification of the Flight-Critical AFTI/F-16 Digital Flight Control System," AIAA-83-0060, Jan. 1983.
11. Mackall, D.A., Regenie, V.A., and Gordo, Michael, "Qualification of the AFTI/F-16 Digital Flight Control System," NAECON Paper 324, May 1983.
12. Holt, H.M., Lupton, A.O., and Holden, D.G., "Flight Critical System Design Guidelines and Validation Methods," AIAA-84-2461, Nov. 1984.
13. Hopkins, Jr., A.L., "General Validation Issues," in "Validation Methods for Fault-Tolerant Avionics and Control Systems — Working Group Meeting I," NASA CP-2114, 1979, pp. 27-30.
14. Hartmann, G.L., Wall, Jr., J.E., and Rang, E.R., "Design Validation of Fly-By-Wire Flight Control Systems," in "Fault-Tolerant Hardware/Software Architecture for Flight Critical Functions," AGARD Lecture Series No. 143, 1985.
15. Gerhart, S.L., "Limitations of Proving and Testing," in "Validation Methods for Fault-Tolerant Avionics and Control Systems — Working Group Meeting I," NASA CP-2114, 1979, pp. 47-53 and 85-88.
16. Petersen, K.L., and Flores, Jr., Christobal, "Software Control and System Configuration Management: A Systems-Wide Approach," NASA TM-85908, 1984.
17. Culbert, Chris, Riley, Gary, and Savely, R.T., "Approaches to the Verification of Rule-Based Expert Systems," SOAR '87, First Annual Workshop on Space Operations Automation and Robotics, Aug. 1987.
18. Richardson, Keith, and Wong, Carla, "Knowledge Based System Verification and Validation as Related to Automation of Space Station Subsystems: Rationale for a Knowledge Based System Lifecycle," in "Conference Proceedings of the Second Annual Artificial Intelligence Research Forum," Information Sciences Division, NASA Ames Research Center, Moffett Field, California, Nov. 1987, pp. 153-158.
19. Stachowitz, R.A., Combs, J.B., and Chang, C.L., "Validation of Knowledge-Based Systems," AIAA-87-1685, Mar. 1987.
20. Duke, E.L., Regenie, V.A., and Deets, D.A., "Rapid Prototyping Facility for Flight Research in Artificial-Intelligence-Based Flight Systems Concepts," NASA TM-88268, 1986.
21. Duke, E.L., Jones, F.P., and Roncoli, R.B., "Development and Flight Test of an Experimental Maneuver Autopilot for a Highly Maneuverable Aircraft," NASA TP-2618, 1986.

VALIDATION OF HIGHLY RELIABLE, REAL-TIME KNOWLEDGE-BASED SYSTEMS

Sally C. Johnson
 System Validation Methods Branch, MS 130
 NASA Langley Research Center
 Hampton, VA 23665-5225

ABSTRACT

Knowledge-based systems have the potential to greatly increase the capabilities of future aircraft and spacecraft and to significantly reduce support manpower needed for the space station and other space missions. However, a credible validation methodology must be developed before knowledge-based systems can be used for life- or mission-critical applications.

Experience with conventional software has shown that the use of good software engineering techniques and static analysis tools can greatly reduce the time needed for testing and simulation of a system. Since exhaustive testing is infeasible, reliability must be built into the software during the design and implementation phases. Unfortunately, many of the software engineering techniques and tools used for conventional software are of little use in the development of knowledge-based systems. Therefore, research at Langley is focused on developing a set of guidelines, methods, and prototype validation tools for building highly reliable, knowledge-based systems.

The use of a comprehensive methodology for building highly reliable, knowledge-based systems should significantly decrease the time needed for testing and simulation. A proven record of delivering reliable systems at the beginning of the highly visible testing and simulation phases is crucial to the acceptance of knowledge-based systems in critical applications.

INTRODUCTION

Highly reliable, real-time knowledge-based systems (KBSSs) have been proposed for many aerospace applications, including space station, manned and unmanned spacecraft, as well as civilian and military aircraft and other life-critical applications. For example, continuous operation of a space station will require extensive, around-the-clock monitoring by large numbers of expert ground control personnel unless some degree of system autonomy is obtained through the use of knowledge-based expert systems. Many of the systems proposed for the space station would result in loss of life if they were to fail during operation. Even when personnel are not involved,

the loss of equipment and/or experiments can be prohibitively expensive. Therefore, these on-board systems must be reliable and validatable. Similarly, a pilot's associate or other advisory system, even if not in direct control of the craft, could only be used if the pilot were confident of its outputs. In many emergency situations, a pilot does not have the time to consider how the system arrived at its conclusion, but must quickly and confidently follow the directions he is given. If this were not the case, then the advisory system would never have been needed in the first place.

A credible validation methodology for highly reliable KBSSs does not exist today. Most current research efforts in verification and validation of KBSSs focus on a rapid-prototyping life cycle, review panels, testing, and development of limited static analysis tools for checking consistency and completeness of a rule base [1]. These techniques are necessary, but alone are not comprehensive enough to validate a system to be used in a life-critical application. Consistency and completeness checking only tests for a limited number of prespecified types of errors. The complexity of the knowledge base in a realistic system makes exhaustive testing impossible. More rigorous validation techniques must be developed.

This paper documents the ongoing research at NASA Langley to develop concepts, guidelines, and methodologies for the validation of KBSSs. The scope of the effort and how Langley's research plan was developed are discussed. The state of the art in validation of conventional software is presented. Characteristics of KBSSs affecting validation are discussed, and how validation of KBSSs differs from conventional software is characterized. The research approach being followed at Langley is then presented, followed by details of the methods, guidelines, and prototype tools being developed. Finally, the expected results from this research project are discussed.

BACKGROUND

The research plan presented in this paper is the culmination of a research effort that began at NASA Langley in 1986 [1 - 4]. The first step was to characterize the potential needs for, and identify current research in, validation of KBSSs

ORIGINAL PAGE IS OF POOR QUALITY

through workshops, classes, and industrial contacts. A research team with varied backgrounds from artificial intelligence, software engineering, and validation was then established. The differences between validation of a conventional software system and validation of a KBS were characterized, and applicability of conventional techniques to KBSs was assessed. The major issues and requirements particular to KBS validation were identified. A number of deficiencies in methods available for KBS validation became apparent, and a preliminary set of tools and methods to be developed were then identified to address those deficiencies.

Concepts, guidelines, methodologies, and supporting tools for the validation and verification of KBSs are to be developed. Because of the lack of available validation methods and the proliferation of KBS development projects, the methods and tools developed will be made available to near-term and mid-term KBS development efforts as soon as practicable. Feedback from these development efforts will provide valuable insight as to the effectiveness and comprehensiveness of the tools and methodologies developed.

The target applications are life-critical KBSs for NASA or military aircraft or spacecraft applications with any one or a combination of rule, frame, or object knowledge representations. Most of the tools and techniques developed will also be useful and cost effective for developing high quality KBSs for applications with less stringent reliability requirements. To keep the development effort feasible and within the bounds of realistic funding expectations, a number of issues will not be addressed, including the following topics: automatic programming, validation of learning, cost/reliability tradeoffs, and validation of advanced hardware architectures. These aspects can only be realistically addressed after significant advances are made in other KBS validation and verification areas.

VALIDATION OF CONVENTIONAL SOFTWARE

The development and validation of reliable conventional software is a major concern within NASA, the Department of Defense, and industry. After many years of research and the development of a new engineering discipline -- Software Engineering -- to address this problem, a number of techniques have been developed. Yet, the discovery of software "bugs" in operational life-critical software is not uncommon [5]. The FAA has not yet certified any civil air transports with flight-critical digital avionics. Thus, the techniques used today for conventional software may actually be inadequate for life-critical applications.

When conventional software is developed for life-critical military or space applications, validation is an ongoing process throughout the life cycle [6]. Limited design tools are available to aid in dividing the problem into a hierarchical set of modules. These modules are developed and tested separately and then integrated. The programmers adhere to strict coding standards and other techniques such as

information hiding that have been found to lead to more reliable code. The developed code is subjected to extensive code walkthroughs and inspections in addition to the static checking provided by sophisticated compilers and other static code analysis tools. Each independently developed module is subjected to extensive testing. The interactions between modules are carefully tested during system integration. The system is then subjected to functional testing and finally simulation.

A system developed for a space application is reviewed periodically throughout the development life cycle by a safety assessment team from NASA to ensure that the procedures discussed above are closely followed [7]. Likewise, the developers of a system for a NASA experimental aircraft must convince a NASA safety team that the system is reliable before flight testing can begin. Similar procedures are followed by the Air Force to ensure adherence to MIL-STD-2167 and by the FAA for civil aircraft systems. The assessment teams typically base their estimates of the reliability of a given system on evidence of rigorous adherence to good software engineering techniques and documentation of traceability to specifications as well as on the absence of serious errors uncovered during testing.

Experience with conventional software has shown that the use of good software engineering techniques and static analysis tools can greatly reduce the time needed for testing and simulation of a system. Errors caught early in the implementation phase are significantly easier and less expensive to correct than those uncovered during the testing phase. The focus of the testing phase should be tuning system performance and promoting confidence about the inherent reliability of the program being tested. Errors caught during this phase should represent the occasional translation and coding errors, not major oversights or misunderstandings of the specifications.

Reliability is a characteristic that must be built into the program from the beginning. Making a poorly written program into a reliable one simply by extensive testing is at least extremely difficult and expensive, if not impossible.

CHARACTERISTICS OF KNOWLEDGE-BASED SYSTEMS

There are two major differences between KBSs and conventional software that affect the validation process -- structure and functionality.

A KBS is divided into some form of knowledge base, which may be rules, frames, procedures, or some other structure or combination of structures, and a reasoning algorithm, such as an inference engine, which operates on the knowledge. This separation of the system into algorithm and data, plus the inherent structuring of the knowledge base may actually aid in the validation process.

Unfortunately, many of the techniques and tools used for conventional software are of little use in the development of KBSs. Researchers are just beginning to develop guidelines for implementing software engineering concepts such as

modularization, information hiding, and structured coding. Development shells and preferred KBS languages such as LISP and Prolog do not support strong typing and other features used in static code analysis, and the compilers do little static checking for errors. Code walkthroughs are less effective for KBSs because each piece of the knowledge is viewed individually and interactions are difficult to conceptualize. The use of new symbolic or parallel architectures significantly compounds the validation problem.

In addition to the above differences attributable to the KBS implementation method, there are further differences caused by the fact that KBSs are often used to implement "expert systems." A KBS is usually expected to have considerably more functionality than would be expected for a conventional software system, especially in the case of an expert system. Much of how the system is to operate is not explicitly known at the start of the project and is to be determined by the knowledge engineer during system development. Expert system applications are typically characterized by the absence of a well-understood algorithm or even well-known performance requirements. The knowledge is often poorly understood and may come from different and even conflicting sources. Access to the expert sources may be limited. A rapid-prototyping development life cycle is used, making traceability of requirements to the code more difficult to ensure. The rapid-prototyping life cycle is not unique to KBSs and is beginning to be studied extensively as an acceptable method for developing conventional software. However, it is still generally recommended that the prototype be discarded or used as a working specification for the development of the real system. Without a well-understood algorithm to follow and with often limited access to the "expert," compiling test cases to assess whether the system is operating "correctly" is usually expensive and difficult. These characteristics have given KBSs a well-deserved reputation for ad hoc, trial-and-error development. Therefore, very rigorous verification of safety will be necessary before a KBS can be certified for use in a life-critical application.

A complete validation methodology must necessarily include guidelines for system development throughout the software life cycle. The rapid prototype scheme of software development, which is very favorable for the development of KBSs, must be accompanied by a specification of the system. The rules used in the prototype represent the knowledge that has been collected about how the system should perform. However, there may be unanticipated interactions between these rules. The system specification should include information about the contents of the knowledge base and deductions that should be possible from it. This "metaknowledge" becomes the basis for the validation effort and should include both "do's"--a specification of what the system should do--as well as "don'ts"--what the system explicitly should not do. Each of these assertions about the system must be classified as to level of criticality--whether failure of the assertion could cause loss of life or property or simply inefficiency or passenger discomfort. Most

applications will contain a mix of assertions of various criticality levels.

The most critical assertions of what the system should and should not do, such as crash the plane, must be verified using rigorous techniques, such as formal verification. The search algorithms employed and their implementations and interactions must also be rigorously verified. This includes verification that the search algorithms will complete within real-time deadlines.

APPROACH

The emphasis on KBS validation research at NASA Langley has been placed on aiding the KBS developer in building a quality product and assessing it before the final phases of testing and simulation are reached. Testing and simulation are then used to assess and tune how well the KBS performs the desired functionality requirements, rather than to try to verify safety properties.

There are several reasons for concentrating research efforts on the design and implementation phases. Two reasons come from experience with conventional software. First, errors are much easier and less expensive to correct if uncovered early in the development life cycle. Also, since exhaustive testing of a nontrivial system is impossible, testing cannot be expected to catch enough errors to change an inherently unreliable program into a reliable one. Most importantly, the largest impediment to deployment of KBS for a life- or mission-critical application is a categorical lack of confidence in all KBSs on the part of those who ultimately make such decisions. This is true of any methods or technologies that are viewed as being radically new and different. The only way to change this image is to arrive at the highly visible testing phase with reliable software and use testing merely to tune system performance. Seeing the uncovering of a number of serious errors during the testing phase of any piece of software alarms safety review teams.

Thus, NASA Langley's efforts in KBS validation research will consist of developing and assessing a number of guidelines and methods for building high reliability into KBSs before they reach the testing phase. The research topics being pursued by NASA Langley and its contractors and grantees are discussed in the following section. Some of the projects discussed below have not even begun yet, and few have progressed past an initial feasibility study phase.

THE PRELIMINARY SET OF TOOLS AND METHODS

A preliminary set of guidelines, methods, and tools have been identified as promising for the development and validation of highly reliable, real-time KBSs. Prototypes of the tools will be developed and integrated with a development environment. The methods and guidelines will be developed, documented, and demonstrated on KBS applications. The preliminary research projects to be pursued include:

- guidelines for scoping the application
- requirements documentation tool
- guidelines for knowledge acquisition
- a development environment supporting software engineering techniques
- consistency and completeness checking tool
- sensitivity analysis tool and guidelines
- methods and tools for formal verification of safety properties
- a base of reasoning algorithms formally characterized to support formal verification
- methods for real-time performance analysis
- methods for implementing a KBS on a fault-tolerant parallel processor

The tools and methods will be applied to several applications, such as the Systems Autonomy Demonstration Project (SADP) demonstration systems, to assess their effectiveness.

Scoping the Application

Before development begins, it is essential to determine a feasible application, or to "scope" the application. This is especially difficult and important for a KBS because of the overzealous selling of AI leading to statements such as "we don't have to know how to do it, we can program it using AI." A set of periodically updated guidelines will be developed for choosing and scoping applications for development. As more KBS development and validation tools and software engineering methods become available, these guidelines will be modified to reflect the current state of the art in KBS development.

Requirements Definition

Validation must be in mind from the beginning of system development. To be useful later in the validation phase, the requirements for the system are divided into the following categories [1]:

1. Desired Competency Requirements -- How well the system is expected to perform. Since the functionality desired from the system is often poorly understood before the system is built, these may of necessity be vague and incomplete.
2. Minimum Competency Requirements -- What the system explicitly must do and must not do to ensure safe operation. These must be precise and comprehensive to support validation and should be rated as to level of criticality.

The requirements developed during this phase and the metaknowledge collected during the knowledge acquisition phase will be documented using a requirements documentation tool. This tool will support traceability between the requirements and the implementation. Also, the consistency and completeness checker and safety property verification tool will directly access this information during the validation phase. Guidelines for developing specifications and guidelines for specification of safety properties will also be developed.

Knowledge Acquisition

A set of guidelines for knowledge acquisition to support validation will be developed. Three types of information should be collected from the experts during the knowledge acquisition phase:

1. Knowledge -- Procedural information about how the system should perform its operation.
2. Metaknowledge -- Metaknowledge, or knowledge about knowledge, describes constraints on the knowledge that can later be used for consistency and completeness checking. The metaknowledge should be documented using the Requirements Documentation Tool.
3. Test cases -- Examples of what proper outputs would be for given inputs to the system.

System Development

The knowledge base is developed from the above information using rapid prototyping on a system development environment, similar to an expert system shell. The system development environment will form the core of the integrated toolset. The development environment must be able to support the development of a KBS composed of a combination of knowledge representations of rules, frames, and objects. The validation and verification tools must be able to directly access the KBS as it is developed. The reasoning algorithm will be chosen from a suite of algorithms or separately developed and formally characterized.

A basic development environment will be chosen from the available environments. The chosen environment will then be enhanced to extend its capabilities, provide support for frames and objects as well as rules, and target it to support probable future NASA applications. Much research will also be done in assessing the software engineering techniques being developed for KBS and in developing new methods such as those used for conventional software, including coding standards for modularization, information hiding, and strong typing. An example of the application of software engineering techniques to KBSs may be found in [8]. Support for these software engineering techniques will be added to the development environments.

Consistency and Completeness Checking

A static analysis tool, including a completeness and consistency checker will be integrated with the toolset to automatically check that the knowledge in the system meets the conditions described by the metaknowledge collected during knowledge acquisition. Quite a number of static analysis tools with various capabilities are currently being developed in industry [9-12]. The Lockheed AI Center has been identified as the source for research and development of a static analysis tool because of their extensive background and sizeable accomplishments in the development of the EVA system.

Current tools are still limited in what they can check for; however, checking of more complex forms

of metaknowledge should be possible in the future. Research will be conducted to assess the usefulness of various types of static KBS analysis. The tool will then be enhanced to provide the types of checking found to be most useful. The static analysis tool will be very useful for finding some types of errors in a knowledge base, but it can only find errors that specifically violate the metaknowledge given. Some verification that the system meets its minimum competency requirements could be done by this tool.

Sensitivity Analysis

Because of the trial-and-error methods often employed in KBS development, KBSs frequently exhibit "instability" or "fragility" properties. These include sensitivities to:

1. Sequence dependencies -- Depending on the order of rule firings, the same input can produce wildly different outputs.
2. Input values -- Slight changes in input values produce extreme changes in output values.
3. Constants -- Slight changes of numerical values contained within the knowledge base, such as constants encoded within the rules or certainty factors, produce extreme changes in output values.

These sensitivities do not necessarily mean that an error is present, but point to likely errors and to values which must be very accurate because the system computation is extremely sensitive to them.

The sensitivity analysis research and tool development is being conducted under a grant to Worcester Polytechnic Institute. A sensitivity analysis tool will be developed to automatically perform specified sensitivity analyses. The tool development is based on the use of Evidence Flow Graphs, which are independent of the knowledge representation of the KBS [3]. A rules-to-graph translator is already being developed to automatically translate a knowledge base of rules from the system development tool into a graphical structure. Other translators to perform translation of other knowledge representations will also be developed. Research will be conducted to extend the types of sensitivity analyses performed and assess their usefulness in finding errors in the knowledge base.

If the system is to be used in a control application, its stability must be validated [13]. Each input value is known only within certain error tolerances. This value is manipulated mathematically using other input values and parameters that also have degrees of error. It must be shown that the result computed by the system is within the tolerance needed by the system. The maximum error of input values and parameters as well as the error tolerances allowable on the outputs must be included in the specification.

Verification of Safety Properties

The KBS must be mathematically verified to meet the minimum competency requirements for safe operation. This is an expensive step, but one that is necessary for life-critical applications. Research into specification of safety properties and mathematical verification of them is being conducted by SRI International. These procedures will be applied to an example application to demonstrate the feasibility of formal verification of safety properties of a realistically complex system. A safety verification tool will be developed to aid the user in this process. The actual mathematical verification will be performed by a theorem prover being developed by SRI for conventional software and hardware [14]. The safety verification tool is basically an interface between the development environment and the theorem prover and will directly access the knowledge base and reasoning control information stored in the development environment.

Reasoning Algorithms

Although many KBSs are written in rules that look like sentences in formal logic, reasoning algorithms typically perform operations that bear no resemblance to first-order logic, such as Prolog's treatment of negation and "cuts." For formal verification of safety properties to be possible, the formal semantics of these features must be defined and adherence of the algorithm used to the defined semantics must be verified. For most applications, one or a combination of several reasoning (inference) algorithms will be chosen from an established base of algorithms. If a new reasoning algorithm must be developed for the application, the semantics of the new algorithm must be defined and verified. Research to develop techniques for semantic characterization of reasoning algorithms, verification of those characterizations, as well as establishment of a base of characterized reasoning algorithms will be performed by SRI International.

Real-time Performance Analysis

Large KBSs are notorious for their very slow performance. Any performance gains expected from the development of faster symbolic processors and more efficient implementations will probably be offset by the growing size and complexity of systems. Because of interactions between knowledge, addition of knowledge to the system can result in exponential increases in search times. Verification that the search algorithms will complete within real-time deadlines will be very important in applications such as aircraft control and advisory systems that have deadlines on the order of a few milliseconds.

A worst case analysis will probably be too conservative to be useful for many systems. However, it may be possible to show analytically that the probability of missing a real-time deadline is within the reliability requirements of the system.

Real-time performance is a function of the hardware architecture, the reasoning algorithm,

how that algorithm is implemented on the hardware architecture, plus the structure and contents of the knowledge base. Techniques for performing real-time performance analysis based on measurable parameters of the system will be developed by the Charles Stark Draper Laboratory.

Parallel Architectures

As KBSs become larger and more complex, the use of parallel architectures will be necessary to obtain acceptable performance. The Charles Stark Draper Laboratory is developing a functional programming model for implementation of a KBS on a fault-tolerant parallel processor. The programming model will provide for graceful degradation, deadlock detection and recovery from excessive generation of parallelism, and distributed checkpointing and error recovery as well as load balancing to increase system performance. Once the programming model is implemented, the system will be used to study optimal KBS parallelization schemes for maximizing performance on a parallel processor and to study real-time performance analysis.

EXPECTED RESULTS

Although none of the tools and methods will be completed in the near term, many of the basic concepts behind those tools and methods are already being developed. Much of the development and validation methodology will be useful to system builders in the near term even before details are worked out and tools are developed. This includes guidelines for what types of information should be collected during the requirements specification and knowledge acquisition phases, how this knowledge can support the validation effort, and various sensitivity analyses to be performed. Guidelines for choosing and scoping a feasible application will have been documented, and a description of software engineering practices that are useful for KBS will have been developed. The first flight test of a simple KBS application, the Mode Control Logic Panel developed by Langley's Aircraft Guidance and Controls Branch, will be conducted in Summer 1988 on the Advanced Transport Operating Systems (ATOPS) aircraft at Langley. This system was developed as a rule-based system then coded in the C programming language.

An integrated prototype toolset with limited validation capabilities should be available for system builders to use by the mid 1990's. The tools and methodologies will be made available to interested KBS developers as beta-test sites, and documentation and consultation on the use of the tools will be made available. Feedback as to the usability and effectiveness of the tools and techniques will be a crucial part of future planning.

CONCLUDING REMARKS

The aim of research at NASA Langley in validation of KBSs is to develop a set of guidelines, methods, and tools to aid a KBS developer in building a highly reliable KBS. An integrated toolset of prototype tools will be developed to demonstrate the methods and how to implement them.

The integrated toolset will in no way be comprehensive enough to support the development of all or even most future NASA KBS applications. The development of a user-friendly toolset with an advanced, comprehensive development environment will be left to industry, but will hopefully be supported by the core research of this project.

The methods and tools being developed purposefully end at the beginning of the testing phase. Exhaustive testing of a realistically complex KBS is impossible. Many KBSs are expected to have considerably more functionality than conventional software and to operate correctly in unanticipated environments. Testing over various expected scenarios will typically uncover only very obvious errors and will not significantly add to the robustness of the KBS or its ability to operate correctly in other unanticipated scenarios. Thus, the system should be relatively reliable before it reaches the testing phase, and testing and simulation should be concentrated on tuning system performance.

REFERENCES

1. Rushby, John, "Quality Measures and Assurance for AI Software," Contract NAS1-17067 Task 5 Final Report, SRI International, Menlo Park, CA., 1988.
2. Morell, Larry, "Use of Metaknowledge in Verification of Knowledge-Based Systems," First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, Tennessee, June 1-3, 1988.
3. Green, Peter, and Becker, Lee, "Evidence Flow Graph Methods for Validation and Verification of Expert Systems," NASA CR to be published.
4. Beaton, Robert, "Reliability and Performance Evaluation of Systems Containing Embedded Rule-Based Expert Systems," NASA CR to be published.
5. Neumann, Peter G., "Some Computer-Related Disasters and Other Egregious Horrors," ACM Software Engineering Notes, Vol. 10, No. 1, January 1985.
6. Spector, Alfred, and Gifford, David, "The Space Shuttle Primary Computer System," Communications of the ACM, Vol. 27, No. 9, September 1984.
7. Culbert, Chris, Riley, Gary, and Savely, Robert, "Approaches to the Verification of Rule-Based Expert Systems," First Annual Workshop on Space Operations, Automation, and Robotics (SOAR87), NASA CP 2491, August 1987.
8. Jacob, Robert J. K., and Froscher, Judith N., "Developing a Software Engineering Methodology for Knowledge-Based Systems," Naval Research Laboratory, Arlington, Va., NRL Report 9019, December 1986.
9. Stachowitz, R. A., Combs, J. B., and Chang, C. L., "Validation of Knowledge-Based Systems," Second AIAA/NASA/USAF Symposium on Automation,

Robotics and Advanced Computing for the
National Space Program, Arlington, Va., March
1987.

10. Bellman, Kirstie L. and Walter, Donald O.,
"Testing Rule-Based Expert Systems," November
1987, submitted for publication.
11. Suwa, Motoi, Scott, A. Carlisle, and
Shortliffe, Edward H., "An Approach to
Verifying Completeness and Consistency in a
Rule-Based Expert System," AI Magazine, Vol.
3, No. 4, Fall 1982.
12. Nguyen, Tin A., Perkins, Walton A., Laffey,
Thomas J., and Pecora, Deanne, "Knowledge Base
Verification," AI Magazine, Vol. 8, No. 2,
Summer 1987.
13. Castore, Glen, "A Formal Approach to
Validation and Verification for Knowledge-
Based Control Systems," First Annual Workshop
on Space Operations, Automation, and Robotics
(SOAR87), NASA CP 2491, August 1987.
14. Moser, Louise, Melliar-Smith, Michael, and
Schwartz, Richard, "Design Verification of
SIFT," NASA CR 4097, September 1987.

Testing Expert Systems

C. L. Chang and R.A. Stachowitz
Lockheed Missiles & Space Company, Inc.
Lockheed Artificial Intelligence Center, O/90-06, B/30E
2100 East St. Elmo Road
Austin, Texas 78744

Abstract

Software quality is of primary concern in all large-scale expert system development efforts. Building appropriate validation and test tools for ensuring software reliability of expert systems is therefore required.

The Expert Systems Validation Associate (EVA) is a validation system under development at the Lockheed Artificial Intelligence Center. EVA provides a wide range of validation and test tools to check the correctness, consistency, and completeness of an expert system.

Testing is a major function of EVA. It means executing an expert system with test cases with the intent of finding errors. In this paper, we describe many different types of testing such as function-based testing, structure-based testing, and data-based testing. We describe how appropriate test cases may be selected in order to perform good and thorough testing of an expert system.

INTRODUCTION

It has been repeatedly shown that the expert system technology in artificial intelligence can be used to implement many different applications such as diagnostic systems, battle management systems, machine and robot control systems, monitoring systems, design systems, manufacturing systems, etc. Regardless of whether the expert systems are stand-alone or real-time embedded systems, we need to be ensured that they are reliable, correct, consistent and complete. For this purpose, the Lockheed Artificial Intelligence Center started in 1986 the Expert Systems Validation Associate (EVA) project [Stachowitz et al. 1987a, 1987b, 1987c]. EVA provides a wide range of validation and test tools to check the correctness, consistency and completeness of an expert system.

Testing is a major function of EVA. It means executing an expert system with test cases with the intent of finding errors. A very good example is the Target Generation Facility (TGF) which provides simulated, real-time controllable aircraft targets to Air Traffic Control Systems under test at the FAA Technical Center.

In this paper, we consider many different types of testing such as function-based testing, structure-based testing and data-based testing. We describe how appropriate test cases may be selected in order to perform good and thorough testing of an expert system.

BACKGROUND

Expert systems are usually developed incrementally. The initial requirements for an expert system may be clearly stated. However, as the expert system evolves and is evaluated, the requirements may be changed or new requirements may be added. In many cases, even if the requirements are not changed, there are no known algorithms for solving the problem. For example, there are no algorithms for performing parallel parking even though the initial and final positions of a car can be specified precisely. Therefore, an expert system may have to be developed in repeated cycles of implementation, evaluation and modification steps. In the parallel parking example, a fuzzy (approximate) algorithm, represented by rules, may be tried first. The algorithm continues to

be modified until a satisfactory performance is achieved. The goal of the test case generator is to generate "appropriate" test cases from the requirements specifications or the expert system itself for users, expert collaborators, or system builders to perform the thorough evaluations during the development or acceptance phase of the system production cycle.

Testing of conventional software [DeMillo et al. 1981, 1987, Hetzel 1984, Miller and Howden 1984, Zeil and White 1980] has been known for a long time. As stated in [Hetzel 1984], software testing is a creative and difficult task. It requires very good knowledge about the system being tested. Typically, the requirements cannot be processed automatically, or knowledge is buried inside the codes of the system. Therefore, test cases are conventionally generated manually. This is certainly tedious and error-prone.

On the other hand, an expert system is usually implemented in a high-level language that supports high-level concepts such as objects, relations, categories, functional mappings, data types and data constraints. This knowledge can be used to generate test cases automatically.

TYPES OF TEST CASES

In this paper, we consider three types of test cases, namely, *function-based test cases*, *structure-based test cases*, and *data-based test cases*.

To generate function-based test cases for an expert system, one requires knowledge about the system's functions. Function-based testing is usually regarded as *black box testing* because it tests the external input-output behavior (specifications) of the system. In order to generate function-based test cases *automatically*, the generator must be provided with knowledge about the input-output specifications.

Structure-based test cases explore the relations between rules. An expert system can be represented by a knowledge base consisting of facts and rules, which can be connected to make a connection graph. An arc in the connection graph denotes a match between a literal in the left hand side (LHS) of a rule and a literal in

the right hand side (RHS) of a rule. Note that a fact can be considered as a rule without a RHS. Structure-based test cases are based upon the structure of the connection graph. The idea is to generate a set of test cases to exercise every rule in the connection graph at least once.

The difference between function-based and structure-based testing can be illustrated by using an electrical circuit: Function-based testing means checking whether the light goes on when we throw the switch, while structure-based testing means inspecting whether all parts are connected properly into the circuit. To perform function-based testing, we do not need to look inside the circuit box. Therefore, it is called *black box* testing. On the other hand, since we need to look inside the circuit box to see how parts are connected, structure-based testing is called *white box* testing.

Data-based test cases are based upon data definitions for the expert system. The data definitions consist of data declarations and data constraints. The data declarations are schema statements for data domains, relations and objects. A data constraint is specified by a logic formula using object-level and/or meta-level predicates.

FUNCTION-BASED TESTING

Input data to an expert system are usually represented by facts that are instances of schemas. Let us call these schemas *input schemas*. Each test case contains a set of facts of the input schemas. For each set of input facts, the expert system will produce a set of output facts (data), which are instance of *output schemas*.

The input and output schemas may not be declared *explicitly*. They may be *implicitly* contained in the connection graph of the expert system. In this case, we consider only the rule part of the connection graph. In a connection graph, there are two kinds of leaf nodes, namely, *input* nodes and *output* nodes. An input node is a LHS-literal of a rule that is not connected to other RHS-literals. An output node is a node representing a RHS-literal that is not connected to any LHS-literals. The schemas of input and output nodes will be considered as the input and output schemas, respectively.

In order to thoroughly cover all different types of input test cases, we must systematically categorize input and output data by explicit declarations. For each set of input facts in certain categories, we specify the expected output facts, or the expected categories of the output facts, or the data constraints that the expected output facts have to satisfy.

Consider the airline inquiry system in [Hetzel 1984]. The specifications of the system are given as follows: The inputs are 1) a transaction identifying departure and destination cities and travel date, and 2) tables of flight information showing flights available and seats remaining. The system checks the flight tables for the desired city. If there is no flight to that city, it prints message 1 "No flight". If there is a flight, but seats are not available, it prints message 2 "Sold out". If there is a flight and seats are available, it displays that fact. Therefore, the expected output is either a flight display, or message 1, or message 2.

For this example, the relational schema is:

```
flight(flight#, from_city, to_city, date, seats_reserved, capacity)
```

where flight# is a key. The data base contains a collection of ground instances (facts) of the flight relation. To generate test cases, we specify the following categories of flights:

```
category(flight,no_flight(X,Y,D)):- /* no flight from X to Y */
    A={F# | flight(F#,X,Y,D,_,_)}, count(A)=0.

category(flight,single(X,Y,D)):- /* single flight from X to Y */
    A={F# | flight(F#,X,Y,D,_,_)}, count(A)=1.

category(flight,multiple(X,Y,D)):- /* multiple flight */
    A={F# | flight(F#,X,Y,D,_,_)}, count(A) > 1.
```

```
category(flight,full(F#,X,Y,D)):- /* flight F# from X to Y is full */
    flight(F#,X,Y,D,S,C), count(S)=C.

category(flight,available(F#,X,Y,D)):- /* flight is available */
    flight(F#,X,Y,D,S,C), count(S) < C.
```

Similarly, the categories of the output on a computer display are:

```
category(output,one_line).
category(output,multiple_lines).
category(output,message_1).
category(output,message_2).
```

(Note that we use the Prolog syntax for representing facts and rules, where a variable is written as a string beginning either with a capital letter or "_".)

For each set of input facts (data) belonging to a certain combination of categories, we specify the expected output. For this example, the input-output relationships specified in terms of categories are given as follows:

- (1) single & available --> one_line.
- (2) multiple & available --> multiple_lines.
- (3) no_flight --> message_1.
- (4) single & full --> message_2.
- (5) multiple & full --> message_2.

Based upon these functional specifications, the test case generator can generate the following test cases to cover different input scenarios:

CASE 1A: Flight available (only flight to the city).

EXPECTED RESULT: Display one line.

CASE 1B: Flight available (multiple flights to the city).

EXPECTED RESULT: Display multiple lines.

CASE 2: No flight.

EXPECTED RESULT: Message 1.

CASE 3A: No seats (only flight to the city).

EXPECTED RESULT: Message 2.

CASE 3B: No seats (multiple flights, all full).

EXPECTED RESULT: Message 2.

CASE 4: Flight available (one flight full, but another open).

EXPECTED RESULT: Display lines and Message 2.

Note that each of CASE 1A through 3B corresponds to one of the input-output relationships specified above. However, CASE 4 is generated by using the input-output relationships (2) and (4). This is possible because the conditions in the input-output relationships (2) and (4) are not mutually exclusive.

STRUCTURE-BASED TESTING

Another important source of test cases derives from the structure of a knowledge base, namely, the connection graph. The advantage of structure-based testing is that the generation of test cases

depends upon only the connection graph. It does not have to rely upon other information such as input-output specification of the system represented by the knowledge base.

The basic concept in structure-based testing is one of *complete coverage*. The assumption is that every rule in the connection graph in some way serves some purpose for handling certain situations. Therefore, all the rules must be useful, i.e., used some time, and the goal of structure-based testing is to generate a set of test cases to exercise every rule at least once. An algorithm for generating such test cases follows:

- (1) Generate the connection graph of the knowledge base.
- (2) Generate the rule flow diagram from the connection graph. Note that a *rule flow diagram* is a directed graph where nodes denote rules, and arcs denote rule execution sequences.
- (3) Create a set of paths in the flow diagram such that each node (rule) is covered by at least one path in the set.
- (4) Generate test cases to traverse these paths.

Consider a rule-based system that computes the grade of a student from his answers to a quiz. The system compares his answers with the expected answers, counts the number of right answers, computes a numerical score, and then records the grade. His answers are represented by *student(Name,Answers)*, and the expected answers are represented by *expect(N_questions, Correct_answers)*. An instance of *student* and an instance of *expect* constitutes an input to the system. The rules for this system are given as follows:

- (1) `grade(Name,Grade):- student(Name,Answers),
 expect(N_questions, Correct_answers),
 right_answers(Answers, Correct_answers, N_rights),
 Ratio is N_rights/N_questions,
 Score is Ratio*100,
 compute_grade(Score,Grade).`
- (2) `right_answers([],[],0).`
- (3) `right_answers([X|Y], [X|Z], R1):-
 right_answers(Y,Z,R),
 R1 is R+1.`
- (4) `right_answers([_|Y], [_|Z], R):- right_answers(Y,Z,R).`
- (5) `compute_grade(Score,a):- Score>=90.`
- (6) `compute_grade(Score,b):- Score<90, Score>=80.`
- (7) `compute_grade(Score,c):- Score<80, Score>=70.`
- (8) `compute_grade(Score,f):- Score<70.`

The rule flow diagram for these rules is shown in Figure 1. From the rule flow diagram, we can construct, for example, a set of paths, [1,3,4,2,8], [1,3,2,5], [1,4,3,3,3,2,7], and [1,3,4,3,3,2,6]. This set has a complete coverage of the rules, because every rule appears in the set at least once. For each path in the set, we collect all the conditions of the rules in the path, and find values that satisfy the conditions. If such values exist, then the path can be traversed, and the values can be used as a test case. The test cases for the paths are shown in Table 1.

DATA-BASED TESTING

We now consider test cases that are derived from data definitions. Such test cases are called *data-based test cases*. Data definitions include data declarations and data constraints. In an expert system shell, data declarations are specified by data schema state-

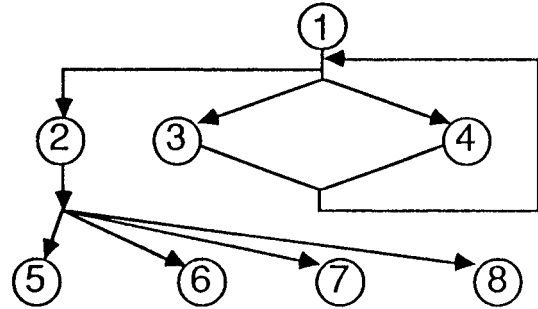


Figure 1. Rule Flow Diagram

TABLE 1. TEST CASES FOR COMPLETE COVERAGE

TEST CASES	PATHS TRAVERSED
student(john, [yes,yes]). expect(2, [yes,no]).	1,3,4,2,8
student(smith, [yes]). expect(1, [yes]).	1,3,2,5
student(peter, [yes,yes,yes,no]). expect(4, [no,yes,yes,no]).	1,4,3,3,3,2,7
student(mary, [yes,no,yes,no,yes]). expect(5, [yes,yes,yes,no,yes]).	1,3,4,3,3,3,2,6

ments. Since maintaining the integrity of facts and rules in a knowledge base is important, we need also to specify the data constraints that the facts and rules must satisfy. Any fact or rule that violates the data constraint will not be inserted into the knowledge base. We can use logical formulas to represent the data constraints.

By means of the data declarations and data constraints in the expert system, we can generate good and bad test cases. A good test case satisfies the data declarations and data constraints and should be accepted by the expert system, while a bad test case violates them and should be rejected by the expert system. Because the goal is to test the expert system with difficult examples, we should generate some extreme cases that barely satisfy or violate the data constraints, or contain large or small values.

Consider input data on triangles specified by

RELATIONAL SCHEMA:

triangle(side1:number, side2:number, side3:number).

DATA CONSTRAINT:

triangle(X,Y,Z) \wedge
 $X + Y > Z \wedge$
 $X + Z > Y \wedge$
 $Y + Z > X.$

The data constraint says that the sum of any two sides of a triangle is greater than the remaining side. From the above data declaration and data constraint, we can generate the following extreme test cases. (Note that the first five test cases are bad, while the last two are good.)

EXTREME TEST CASES	COMMENTS
triangle(1, 1, 2)	A straight line
triangle(0, 0, 0)	A point
triangle(4, 0, 3)	A zero side
triangle(1, 2, 3.00001)	Close to a triangle
triangle(9170, 8942, 1)	Very small angle
triangle(.0001, .0001, .0001)	Very small triangle
triangle(83127, 74326, 96652)	Very large triangle

For an *applicative* system which takes an input and produces an output, a test case means a simulated instance of input and its expected output. However, for an *imperative* system that may alter data structures or produce side effects, just generating test cases of input is not enough. An imperative system can be represented by a state machine. There are a number of states. For each state, there are a certain number of actions that take the state into other states. For the state machine, a test case will be actually a test scenario that consists of an initial state, and a sequence of specific actions. The goal is to check if bad states will be encountered when we run the state machine with the test scenario. We note that a bad state means that the state violates integrity constraints or a situation where no actions are available.

Consider the following example: Container A can hold 5 gallons of water and container B 2 gallons of water. Initially, A is full and B is empty. Assume that water can be poured from A to B, and B to the drain. We would like to get to a final state where A is empty and B is half-full. The initial and final states are shown in Figure 2.

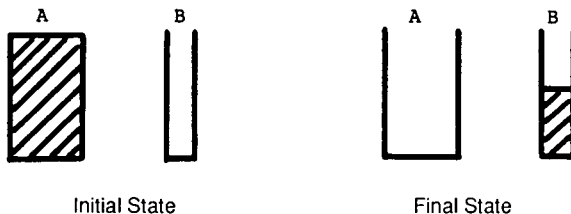


Figure 2. Initial and Final States

We use $state(X,Y)$ to denote a state where X and Y are the amounts of water in containers A and B, respectively, and use $pour(X,Y,Q)$ to denote an operation to pour Q gallons of water from X into Y .

Let $transition(Op,X,Y)$ denote that the operation Op changes state X to state Y , and let $reach(Seq,X,Y)$ denote that the sequence of operations, Seq , changes state X to state Y .

The constraints on states and operations are specified as follows:

$pour(a,b,X) \wedge 0 < X \leq 2$.

$pour(b,drain,X) \wedge 0 < X \leq 2$.

$state(X,Y) \wedge 0 \leq X \leq 5 \wedge 0 \leq Y \leq 2$.

From the constraints, we can generate the following test scenario:

$state(5,0)$. initial state
 $pour(a,b,2), pour(a,b,2)$. input sequence

This is a bad test scenario because the second operation in the input sequence will cause container B to overflow. If we had the following knowledge base,

(1) $transition(pour(a,b,Z), state(X,Y), state(U,V)) :-$

$Z > 0 \wedge$

$U = X - Z \wedge$

$V = Y + Z$.

(2) $transition(pour(b,drain,Y), state(X,Y), state(X,0)) :- Y > 0$.

(3) $reach([Op], S1, S2) :- transition(Op, S1, S2)$.

(4) $reach([Op|Seq], S1, S3) :-$

$transition(Op, S1, S2),$

$reach(Seq, S2, S3)$.

the bad scenario would be "successfully" processed, because Rule (1) is wrong. The correct version of the rule should be

(1') $transition(pour(a,b,Z), state(X,Y), state(U,V)) :-$

$Z > 0 \wedge$

$U = X - Z \wedge$

$V = Y + Z \wedge$

$X \geq Z \wedge$

$V \leq 2$.

The correct rule does make sure that container B will not overflow.

CONCLUSION

Test cases of input values to an expert system can be generated automatically. However, the expected output and performance for each test case may not be known, or not clearly defined, or stated in qualitative or narrative statements. In this case, the system's output and performance for the generated (simulated) test cases may have to be evaluated by independent human experts. The experts' evaluation results can be stored and used with the test cases again when the expert system is modified.

We have described systematic ways for automatic test case generation. For large expert systems, this is essential because manual approaches are tedious and possibly biased.

We have started work on implementing components of the test case generator. First, we will generate structure-based test cases because they do not depend upon specifications and metaknowledge. Then, we will consider data-based and finally function-based test cases.

REFERENCES

- Bellman, K.L., and Walter, D.O. [1987] "Testing rule-based expert systems", Technical Report, Knowledge-Based Systems Section, Computer Science Laboratory, The Aerospace Corporation, Los Angeles, Ca 90009-2957, November 1, 1987.
- Chang, C.L. [1976] "DEDUCE --- A deductive query language for relational data bases", in *Pattern Recognition and Artificial Intelligence* (C.H. Chen, Ed.), Academic Press, Inc., New York, 1976, pp.108-134.
- Chang, C.L. [1978] "DEDUCE 2: Further investigations of deduction in relational data bases", in *Logic and Data Bases* (H. Gallaire and J. Minker, Eds.), Plenum Publishing Corp., New York, 1978, pp.201-236.
- Chang, C.L. [1981] "On evaluation of queries containing derived relations in a relational data base", in *Advances in Data Base Theory --- Volume 1* (H. Gallaire, J. Minker and J.M. Nicolas, Eds.) Plenum Publishing Corp., 1981, pp.235-260.
- Culbert, C., Riley, G., and Savely, R.T. [1987] "Approaches to the verification of rule-based expert systems", *Proc. of First Annual Workshop on Space Operations Automation and Robotics*, Houston, Texas, August 1987.
- DeMillo, R.A., Hocking, D.E., and Merritt, M.J. [1981] "A comparison of some reliable test data generation procedures", GIT-ICS-81/08, School of Information and Computer Science, Georgia Institute of Technology, April 1981.
- DeMillo, R.A., McCracken, W.M., Martin, R.J., and Passafiume, J.F. [1987] *Software Testing and Evaluation*, The Benjamin/Cummings Publishing Company, Inc., 2727 Sand Hill Road, Menlo Park, California 98025.
- Geissman, J.R., and Schultz, R.D. [1988] "Verification and validation of expert systems", *AI Expert*, February 1988, pp.26-33.
- Green, C.J.R., and Keyes, M.M. [1987] "Verification and validation of expert systems", *IEEE Knowledge-Based Engineering & Expert System* (WESTEX-87), IEEE 87CH2463-8, 1987, pp.38-43.
- Hetzel, W. [1984] *The Complete Guide to Software Testing*, QED Information Sciences, Inc., Wellesley, Massachusetts, 1984.
- Miller, E.F. [1987] "Expert system validation: Issues and approaches", in *Expert Systems and Their Applications*, May 1987, Avignon, France.
- Miller, E.F., and Howden, W. [1984] *Software Testing and Validation Techniques*, 2nd Edition, IEEE Computer Society Press, 1984.
- Jacob, R.J.K., and Froscher, J.N. [1986] "Developing a software engineering methodology for knowledge-based systems", NRL Report 9019, Computer Science and Systems Branch, Information Technology Division, Naval Research Laboratory, Washington, D.C. 20375-5000.
- Nguyen, T.A. [1987] "Verifying consistency of production systems", *Proc. of the 3rd IEEE Conference on AI Applications*, February 1987, pp.4-8.
- Nguyen, T.A., Perkins, W.A., Laffey, T.J., and Pecora, D. [1985] "Checking an expert system's knowledge base for consistency and completeness", *Proc. of the 9th International Joint Conference on Artificial Intelligence*, 1985, pp.375-378.
- Stachowitz, R.A., and Combs, J.B. [1987a] "Validation of expert systems", *Proc. of the 20th Hawaii International Conference on Systems Sciences*, 1987, pp.686-695.
- Stachowitz, R.A., Combs, J.B., and Chang, C.L. [1987b] "Validation of knowledge-based systems", *Proc. of the 2nd AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program*, Arlington, Virginia, March 9-11, 1987.
- Stachowitz, R.A., Chang, C.L., Stock, T., and Combs, J.B. [1987c] "Building validation tools for knowledge-based systems," *Proc. of First Annual Workshop on Space Operations Automation and Robotics*, Houston, Texas, August 1987.
- St. Johanser, J.T., and Harbidge, R.M. [1986] "Validating expert systems: Problems & solutions in practice", *Proc. of the International Conference on Knowledge-Based Systems*, London, England, pp.215-229.
- Suwa, M., Scott, A.C., and Shortliffe, E.H. [1982] "An approach to verifying completeness and consistency in a rule-based expert system", *The AI Magazine*, 1982, pp.16-21.
- Weiss, S.M., and Kulikowski, C.M. [1983] "Testing and evaluating expert systems", Chapter 6 in *A Practical Guide to Designing Expert Systems*, Chapman and Hall, pp.138-156.
- Zeil, S.J., and White, L.J. [1980] "Sufficient test sets for path analysis testing strategies", OSU-CISRC-TR-80-6, The Computer and Information Science Research Center, The Ohio State University, Columbus, Ohio 43210, July 1980.

THE AUTHORS

Dr. Chang received his Ph.D. in Electrical Engineering from the University of California, Berkeley, CA in 1967. His background includes design and development of large-scale knowledge-based systems, and research in program generation, very high level languages, compilers, rapid prototyping, relational data bases, natural language query systems, mechanical theorem proving, and pattern recognition. He wrote two books "Symbolic Logic and Mechanical Theorem Proving" (with Dr. Richard Lee), and "Introduction to Artificial Intelligence Techniques", and published more than 50 papers. He is currently a co-principal investigator of the Knowledge-Based Systems Validation project at the Lockheed AI Center.

Dr. Stachowitz received his Ph.D. in Linguistics from the University of Texas at Austin in 1969. His background includes design and development of a large-scale knowledge-based mechanical translation system, computer hardware and software performance evaluation, and research in applicative programming languages, semantic data models, and analytic modeling and performance evaluation of data base machine architectures. He also has performed research in logic and functional knowledge base manipulation and query languages. He is a Senior Research Scientist at Lockheed's Artificial Intelligence Center and co-principal investigator of the Knowledge-Based Systems Validation project.

Diagnosis by Integrating Model-Based Reasoning with Knowledge-Based Reasoning

Tom Bylander

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210
byland@cis.ohio-state.edu

Abstract

Our research investigates how observations can be categorized by integrating a qualitative physical model with experiential knowledge. Our domain is diagnosis of pathologic gait in humans, in which the observations are the gait motions, muscle activity during gait, and physical exam data, and the diagnostic hypotheses are the potential muscle weaknesses, muscle mistimings, and joint restrictions. Patients with underlying neurological disorders typically have several malfunctions. Among the problems that need to be faced are: the ambiguity of the observations, the ambiguity of the qualitative physical model, correspondence of the observations and hypotheses to the qualitative physical model, the inherent uncertainty of experiential knowledge, and the combinatorics involved in forming composite hypotheses. Our system divides the work so that the knowledge-based reasoning suggests which hypotheses appear more likely than others, the qualitative physical model is used to determine which hypotheses explain which observations, and another process combines these functionalities to construct a composite hypothesis based on explanatory power and plausibility. We speculate that the reasoning architecture of our system is generally applicable to complex domains in which a less-than-perfect physical model and less-than-perfect experiential knowledge need to be combined to perform diagnosis.

The Promise of Deep Knowledge

Recently in knowledge-based systems research, there has been an emphasis on "deep" knowledge over "shallow" knowledge. Deep knowledge is based on the causal mechanisms underlying the domain, typically obtained through scientific studies and incorporated into physical models of the domain, while shallow knowledge is based on experiential knowledge, typically obtained from human experts and incorporated in rule-based systems [4]. The promise of deep knowledge is that the conclusions of a knowledge-based system be inferred from an accurate physical model of the domain, rather than dependent on a time-consuming and error-prone knowledge engineering effort.

The emphasis on deep knowledge has influenced research on diagnosis [9, 19]. In this line of research, the process of diagnosis is reduced to manipulating the physical

model within the knowledge-based system, according to the following presumptions.

1. Let M be the physical model which describes the domain when everything is functioning as it should.
2. Let \mathcal{M} be the set of all physical models consistent with the observations. If $M \notin \mathcal{M}$, then there is a malfunction.
3. If there is a malfunction, then for each $M' \in \mathcal{M}$, the differences between M' and M is a possible diagnosis.

That is, the normal physical model is selectively changed until it predicts (or is compatible with) the aberrant observations. Each change corresponds to an abnormality or malfunction. For example, in de Kleer and Williams' method for diagnosis, such a change consists of suspending the constraints of a component in the model, i.e., the outputs of a malfunctioning component are considered to be unconstrained. Each set of changes that accounts for the observations is considered a possible diagnosis. The process of diagnosis is a search for each such set of changes.

An example might clarify the intended role of physical models. Consider the device in Figure 1, by now a familiar example in the literature. Mult-1, Mult-2, and Mult-3 are three multipliers, with A-E as their inputs and X-Z as their outputs. Add-1 and Add-2 are adders with X-Z as their inputs. Add-1 and Add-2 are adders with X-Z as inputs and F and G as outputs. Given specific inputs to this de-

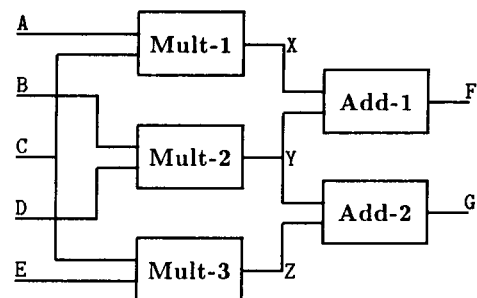


Figure 1: Multiplier and Adder Example

vice, and assuming that the device is functioning properly, the outputs can be predicted. If the actual outputs are improper, malfunctions can be attributed to one or more components by tracing through the connections and determining whether abnormal behavior would account for the outputs. For example, if output F is improper, then one possibility is that Mult-1 has a malfunction because if Mult-1 was no longer behaving like a multiplier, then F would be affected. Further testing with different inputs can verify or rule out this hypothesis about Mult-1 [6, 11].

Note that in this example, experiential knowledge is not mentioned, nor is it needed. The model of the device—the behavior of the components and the interconnections between the components—provides sufficient knowledge for diagnosis to be efficiently performed. If models with similar properties could be constructed for other domains, then the knowledge acquisition bottleneck could be avoided, and the conclusions of knowledge-based systems would be better justified.

Difficulties in Fulfilling the Promise

We have been studying diagnosis in the domain of human pathologic gait (walking disorders) [13, 18]. In this domain, the goal of diagnosis is to determine the muscular and skeletal causes of the patient's abnormal gait motions. Examples of malfunctions include reduced ranges of motion of joints (contractures), joint pain, muscle weakness, and muscle spasticity. Typically, patients will have a known underlying disorder such as cerebral palsy or arthritis, which gives rise to the joint and muscle malfunctions. Diagnosing joint and muscle malfunctions is done to help determine what kind of treatments (e.g., physical therapy, braces, surgery on joints and muscles) will best correct the patient's gait.

The data is primarily of the following types: history, physical exam, and motion data. The patient's history includes information about past and present diagnoses and treatments and demographic data. The physical examination provides data about the range of motion of joints and strength of muscles. The motion data include the gait motions of the patient obtained through a special camera system. This measures joint angles, gait velocity, stride length, etc. Also, EMG measurements are taken while the patient is walking, which provides data on muscle activity.

Since human walking is subject to the laws of Physics, just as any other physical activity, it appears that deep-knowledge-diagnosis would be appropriate. Unfortunately, there are several difficulties in doing diagnosis based on physical models.

1. *Construction.* Domain models with sufficient predictive and explanatory power need to be constructed before deep-knowledge-diagnosis can proceed. However, quantitative modeling of human gait is still a challenging research topic [12]. This is not just a problem in our domain. More often than not, simulation of complex mechanical devices and biological processes are *open research problems*.

2. *Ambiguity.* Even if a domain model can be constructed, there is a problem of obtaining sufficiently detailed data for the domain model. If a quantitative simulation is to be performed, then precise measurements of the initial state and input parameters need to be obtained. In many domains, this presents no difficulties. For example, electronic circuits are sufficiently constrained and well-understood so that carefully selected measurements can give the state of the device. Unfortunately in gait analysis, many internal gait parameters cannot be directly or even indirectly measured by current technology, e.g., EMG data is a best a qualitative measure of muscle forces [20]. Generally in medical domains, many internal parameters cannot be accurately measured without overly invasive actions.

One answer to this problem (and part of our own solution) is the use of qualitative physical models [1, 8, 10, 15]. Such models still provide explanatory power in spite of ambiguous data. However, qualitative models introduce their own sources of ambiguity. As a rule, qualitative simulation does not predict a single sequence of states, but produces several alternative state sequences. Additional information is required to disambiguate between them [7]. Also, qualitative simulation might produce state sequences that are spurious [16].

3. *Computational complexity.* Diagnosis is inherently computationally complex. The number of possible diagnoses is combinatorially large. If n different malfunctions can occur, then there are 2^n possible sets of malfunctions. If each malfunctions can be caused in m different ways, then there are 2^{mn} possibilities. This large hypothesis space is not just an abstract possibility. In pathologic gait, patients with underlying neurological disorders typically have several malfunctions, some of which are "primary" (due to the underlying disorder), while others are attempted compensations.

Clearly, there is a need to modify the assumption that physical models for performing efficient and accurate diagnosis can be readily constructed in all domains. A more reasonable assumption is that physical models can perform specific diagnostic *subtasks*. Experiential knowledge acquired from human experts is still needed to help guide the search through the hypothesis space.

The Subtasks That Physical Models and Experiential Knowledge Are Good For

The next question is to identify the respective roles that physical models and experiential knowledge can play. Unfortunately, many factors are domain-dependent, and no "task model" is sufficiently developed to clearly answer this question (see [3, 5, 17] for what has been developed). Our own experience (the next best thing) is the following.

As mentioned earlier, deep knowledge is usually associated with physical models, which are intended to have pre-

dictive and explanatory power. A physical model of pathologic gait needs to represent at least the muscles, the limb segments, and the interactions among them. Given a particular situation (joint positions, limb and trunk momentums, muscle and ground forces), it ideally should predict changes in position and momentum. A major difficulty is obtaining accurate data on muscle forces. As a consequence, we will rely on a *qualitative* physical model, based on knowledge about the motions that muscles control and on relative strengths of muscles. For example, the action of a muscle on a joint might be described as "causes flexion" as opposed to a differential equation. Such models have weak predictive capabilities. In our domain, we will at best be able to explain how a motion could be caused by a combinations of factors, but the ambiguity concerning the exact amount of force associated with each factor precludes even qualitatively accurate predictions.

Another difficulty is using the physical model to search for diagnostic hypotheses. Because the interactions of the components (muscles and limb segments) are highly complex (unlike the device of Figure 1), a particular motion could be caused in a large number of ways, especially when combinations of malfunctions are considered. Also, the effect of any single malfunction can propagate throughout the rest of the system. A sprained ankle, for instance, affects the whole gait, not just the motion of the ankle. A physical model then might be able to suggest *local, individual* causes for a particular abnormal motion, but searching for all possible causes for each abnormal motion and generating composite hypotheses based on just this information will be computationally prohibitive.

Can experiential knowledge be used for the diagnostic reasoning that is difficult to do with physical models? As is typical in knowledge-based systems, experience can provide rules that associate abnormal observations with the malfunctions that typically cause them. Hence, experiential knowledge can give valuable clues concerning what malfunctions should be considered. Such knowledge, though, is not very good for determining whether a hypothesized malfunction accounts for the observations in a particular case, and is no good for considering combinations of interacting malfunctions.

Thus, we can usually use (qualitative) physical models to suggest some of the possible causes of an observation and to determine what observations a hypothesis accounts for (explanatory coverage). Experiential knowledge can associate hypotheses with observations and suggest which hypotheses appear more likely than others. In general, reasoning based on experiential knowledge is good for generating individual malfunctions that appear likely, while model-based reasoning is best for testing explanatory coverage of malfunctions and combinations thereof. Figure 2 summarizes these conclusions.

Integrating Model-Based Reasoning with Experiential Reasoning

These considerations lead us to the following proposal based on using hypothesis assembly [14] to integrate model-based

Experiential Reasoning	Model-Based Reasoning
generate: malfunctions associated with observations	generate: malfunctions that cause observations
test: likelihood of malfunctions	test: explanatory coverage of malfunctions
good for: generating individual malfunctions	good for: testing combinations of malfunctions

Figure 2: Experiential vs. Model-Based Reasoning

reasoning with experiential reasoning. Hypothesis assembly is a general technique for constructing and critiquing composite hypotheses. It requests information via the following domain-dependent functions:

1. a function that rates the "plausibility" of a hypothesis. The likelihood of malfunctions as determined by experiential reasoning can be used to rate plausibility. Explanatory coverage can be used to break ties (i.e., when the likelihoods of two malfunctions are too close for meaningful comparison).
2. a function that rates the importance of observations. In pathologic gait, the amount of the difference between abnormal and normal for a particular motion parameter determines its importance.
3. a function that determines what hypotheses can explain an observation. The qualitative physical model can be used to suggest hypotheses that explain an observation.
4. a function that determines what observations a (composite) hypothesis does and does not explain. The qualitative physical model can be used to determine what observations are explained by a combination of malfunctions.

The subtasks that experiential knowledge and physical models are good for fit into hypothesis assembly quite well.

Hypothesis assembly uses this information to construct a composite hypothesis with the following properties:

- The composite hypothesis explains as many observations as possible in comparison with similar composite hypotheses. That is, no local change (addition/deletion of some part to/from the composite hypothesis) improves explanatory coverage.
- Each hypothesis part within the composite hypothesis is as plausible as possible, viz. in comparison to other hypothesis parts explaining some particular observation.
- The composite hypothesis is parsimonious, i.e., no hypothesis in the composite hypothesis is superfluous.

(A hypothesis within a composite hypothesis is superfluous if it can be removed without loss of explanatory coverage.)

Hypothesis assembly also critiques this composite hypothesis in comparison to other composite hypotheses. Thus, one composite hypothesis is selected and its goodness in comparison to other hypotheses is determined.

Hypothesis assembly, however, is not guaranteed to find the "correct" hypotheses. Given the difficulties in deep-knowledge-diagnosis discussed earlier, no method can be expected to guarantee truth. Nor is hypothesis assembly guaranteed to produce the "best" hypotheses according to normative criteria such as "most probable hypothesis that accounts for all the observations." Such criteria are computationally intractable [2]. We conjecture that hypothesis assembly is the best that can be done within the constraints of imperfect physical models and computational tractability.

Conclusion

It has been proposed that diagnosis should be based on physical models of the domain. However, several factors make it unlikely that diagnosis can be just be based on physical models. These factors include constructing a sufficiently powerful physical model, obtaining sufficiently accurate observations, and performing diagnosis efficiently. Diagnosis in the domain of human pathologic gait illustrates these problems. Our proposal is to integrate qualitative physical models with experiential knowledge so that both sources of information will be efficiently and effectively utilized. In particular, they can be integrated using the technique of hypothesis assembly, which constructs a composite diagnostic hypotheses with several desirable properties: explanatory power, plausibility, and parsimony. We speculate that the reasoning architecture of our system is generally applicable to complex domains in which a less-than-perfect physical model and less-than-perfect experiential knowledge need to be combined to perform diagnosis.

Acknowledgments

This research has been supported by grants 82048-02 and H133E80017 from the National Institute on Disability and Rehabilitation Research.

References

- [1] T. Bylander. A critique of qualitative simulation from a consolidation viewpoint. *IEEE Trans. Systems, Man, and Cybernetics*, 18(2):252-263, 1988.
- [2] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. *Some Results Concerning the Computational Complexity of Abduction*. Technical Report, Lab. for AI Research, CIS Dept., Ohio State U., Columbus, OH, 1988.
- [3] B. Chandrasekaran. Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. *IEEE Expert*, 1(3):23-30, 1986.
- [4] B. Chandrasekaran and S. Mittal. Deep versus compiled knowledge approaches to diagnostic problem-solving. *Int. J. Man-Machine Studies*, 19:425-436, 1983.
- [5] W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27(3):289-350, 1985.
- [6] R. Davis. Diagnostic reasoning based on structure and function. *Artificial Intelligence*, 24:347-410, 1984.
- [7] J. de Kleer and J. S. Brown. Assumptions and ambiguities in mechanistic mental models. In *Mental Models*, pages 155-190, Lawrence Erlbaum, Hillsdale, NJ, 1983.
- [8] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7-83, 1984.
- [9] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97-130, 1987.
- [10] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85-168, 1984.
- [11] M. R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411-436, 1984.
- [12] H. Hemami. Modelling, control, and simulation of human movement. *Critical Reviews in Biomedical Engineering*, 13(1):1-34, 1985.
- [13] V. T. Inman, H. J. Ralston, and F. Todd. *Human Walking*. Williams & Wilkins, Baltimore, 1981.
- [14] J. R. Josephson, B. Chandrasekaran, J. W. Smith, and M. C. Tanner. A mechanism for forming composite explanatory hypotheses. *IEEE Trans. Systems, Man and Cybernetics*, 17(3):445-454, 1987.
- [15] B. J. Kuipers. Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence*, 24:169-203, 1984.
- [16] B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289-338, 1986.
- [17] J. McDermott. *Preliminary Steps Toward a Taxonomy of Problem Solving Methods*. Technical Report, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, 1988.
- [18] J. Perry. Normal and pathologic gait. In *Atlas of Orthotics*, chapter 4, C. V. Mosby, St. Louis, 1985.
- [19] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57-95, 1987.
- [20] S. R. Simon. Kinesiology—its measurement and importance to rehabilitation. In V. L. Nickel, editor, *Orthopedic Rehabilitation*, chapter 5, Churchill Livingstone, New York, 1982.

**APPLICATION OF CASE-BASED REASONING:
DESIGN FOR MANUFACTURABILITY**

**James A. King
NCR Corporation**

**Leslie Whitaker
Klein Associates**

(Paper not provided by publication date.)

Transformation Based Endorsement Systems

Thomas Sudkamp
Department of Computer Science
Wright State University
Dayton, Ohio 45345

Abstract: Evidential reasoning techniques classically represent support for a hypothesis by a numeric value or an evidential interval. The combination of support is performed by an arithmetic rule which often requires restrictions to be placed on the set of possibilities. These assumptions usually require the hypotheses to be exhaustive and mutually exclusive. Endorsement based classification systems represent support for the alternatives symbolically rather than numerically. A framework for constructing endorsement systems is presented in which transformations are defined to generate and update the knowledge base. The interaction of the knowledge base and transformations produces a non-monotonic reasoning system. Two endorsement based reasoning systems are presented to demonstrate the flexibility of the transformational approach for reasoning with ambiguous and inconsistent information.

1 Introduction

Classification systems are designed to determine the identity of an object from a set of possibilities. Evidence is acquired and interpreted to provide support for the alternatives. Historically, numeric measures have been used to represent the support for the alternatives. Common numeric systems for combining evidential support include certainty factors [2, Chapters 10-11], Bayesian probability and the Dempster-Shafer theory of evidential reasoning [5]. Endorsement based reasoning was introduced by Cohen [3] and Sullivan and Cohen [6] to provide a framework for the symbolic representation and combination of evidential support.

Numeric representations of support, in which the likelihood of a possibility is often indicated by a single value or by an evidential interval, have distinct computational advantages. The combination of support is accomplished by a straightforward arithmetic calculation such as Bayes' rule or Dempster's rule. Moreover, a ranking of the likelihood of the alternatives can be obtained directly from the associated values.

This research was supported by the Air Force Office of Scientific Research under contract FY1175-87-04878/01.

The disadvantages associated with the numeric representation of support have been well chronicled. Difficulties with the use of probabilistic techniques for evidential reasoning are presented in Tversky and Kahneman [7] and Quinlan [4]. Shafer [5] discusses the inadequacy of a single point measure for representing evidential support. Experiments by Buchanan and Shortliffe [2, Chapter 10] exhibit the lack of sensitivity in system performance to changes in the numeric values. The standard numeric techniques also fail when presented inconsistent information. When this occurs, the result of the computation of both Bayes' rule and Dempster's rule is undefined.

An endorsement based system uses symbolic interpretations of the information, endorsements, to represent and combine evidential support. Rather than translating the evidence into a form suitable for a predefined combination rule, the combination techniques are specifically designed for the evidential information of the particular domain. Ranking the alternatives requires an analysis of the endorsements in the knowledge base. Separating the evaluation of the alternatives from the support combination techniques adds flexibility to the reasoning system. It is this separation that permits endorsement based systems to develop hypotheses from inconsistent information.

2 Ambiguity and Inconsistency

Many classification problems can be formulated as questions of the propagation of support in a hierarchy. The relationships of the hierarchy are defined by set inclusion. The alternatives are distinguished by the presence or absence of certain characteristics. For example, a medical diagnosis system attempts to identify a disease from the information provided by the observed symptoms and test results. The diseases comprise the set of possibilities and the characteristics are the symptoms. For identification purposes, a disease is completely characterized by its symptoms.

Formally, a classification hierarchy is defined by two sets; the characteristics C and possibilities P . A possibility is defined as a subset of characteristics. A simple hierarchy is illustrated in *Figure 1*. Throughout this paper, variables

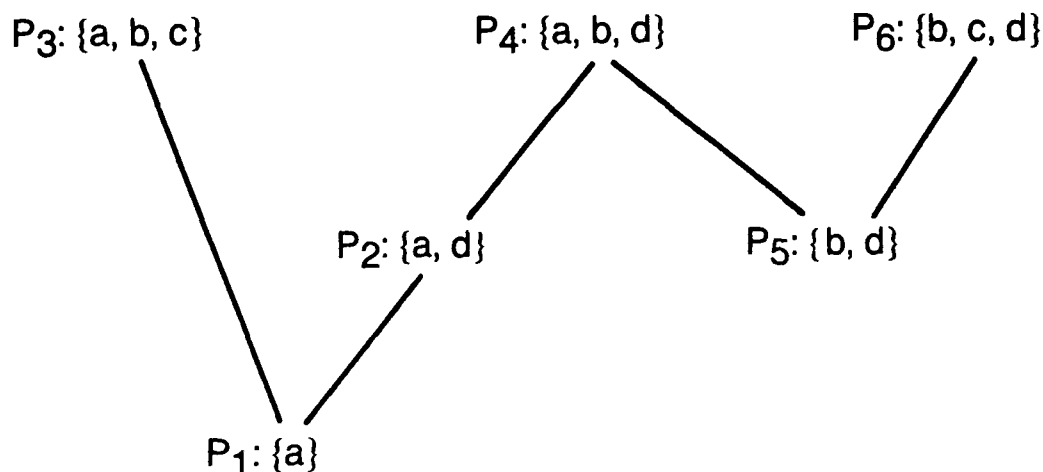


Figure 1. A characteristic, possibility hierarchy.

x and y are used to denote characteristics while X and Y denote possibilities.

Evidence supporting the presence of characteristic a and the absence of d in the hierarchy defined in Figure 1 is *ambiguous* since both P_1 and P_3 are consistent with this information. The addition of evidence supporting the presence of c produces unambiguous evidence; P_3 is the sole consistent possibility. Finally, acquiring information that denies the presence of b provides an example of *hierarchical inconsistency*; there are no possibilities that agree with the accumulated data.

3 A Transformation System

The use of transformation rules to define the combination of support in an endorsement based system is demonstrated by the system GET (Generation of Endorsements by Transformations). The objective is to identify a possibility by acquiring information pertaining to the presence or absence of characteristics. The transformations that define the support combination techniques assert and delete endorsements. The set of asserted endorsements is referred to as the knowledge base. The endorsements for in the system GET are given in Table 1.

Evidence supporting the presence of a characteristic x is denoted $p(x)$, $a(x)$ denotes evidence that indicates the absence of characteristic x . Because of the simplicity of the evidential information, several important capabilities of the transformational approach are not exhibited in this system. Extensions to this basic model are described in Sections 4 and 5.

GET utilizes two types of endorsements; *evidential* and *derived*. Evidential endorsements m , n and d are generated directly from the evidence and the relationships that

define the hierarchy. The endorsement $m(x, Y)$ is asserted whenever evidence $p(x)$ is processed and x is a characteristic of Y . The evidential endorsement $n(x, Y)$ is asserted when evidence is obtained that indicates the presence of a characteristic not in Y . Similarly, $d(x, Y)$ is added to the knowledge base when evidence is obtained indicating the absence of x and x is a characteristic of Y . The endorsement $m(x, Y)$ offers positive support for the possibility Y . The endorsements $n(x, Y)$ and $d(x, Y)$ are negative, they indicate a disagreement between the evidence and the composition of Y .

Derived endorsements are produced by the transformations that define the combination of support. The derived endorsements of GET are s , c , o , and i . These endorsements indicate the consistency of a possibility with the accumulated evidence and are similar to those used by Sullivan and Cohen [6] for recognizing plans.

The knowledge base is maintained by transformations that insert and delete endorsements. The transformations are maintained by two types of rules; *replacement rules* and *generation rules*. Endorsements are generated by rules of the form

$$\text{condition} \Rightarrow \text{endorsement}$$

where the condition may refer to the evidence and to endorsements in the knowledge base. When the condition is satisfied, the rule adds the endorsement to the knowledge base. The generation rules are applied only when the endorsement on the right-hand side is not currently asserted. Consequently, the knowledge base will not contain duplicate endorsements.

A replacement rule has the form

$$e_1, \dots, e_n \rightarrow f_1, \dots, f_k$$

endorsement	interpretation
$m(x, Y)$	x , whose presence is supported, is a member of Y
$n(x, Y)$	x , whose presence is supported, is not a member of Y
$d(x, Y)$	x , whose absence is supported, is a member of Y
$s(Y)$	Y is the sole consistent possibility
$c(Y)$	Y is consistent
$o(Y)$	Y is only one of several consistent possibilities (other consistent possibilities)
$i(Y)$	Y is inconsistent

generation rules	replacement rules
1. $p(x) \& member(x, Y) \Rightarrow m(x, Y)$	8. $s(Y), c(Y) \rightarrow s(Y)$
2. $p(x) \& \neg member(x, Y) \Rightarrow n(x, Y)$	9. $s(Y), o(Y) \rightarrow s(Y)$
3. $a(x) \& member(x, Y) \Rightarrow d(x, Y)$	10. $i(Y), s(Y) \rightarrow i(Y)$
4. $n(x, Y) \Rightarrow i(Y)$	11. $i(Y), c(Y) \rightarrow i(Y)$
5. $d(x, Y) \Rightarrow i(Y)$	12. $i(Y), o(Y) \rightarrow i(Y)$
6. $c(Y) \& \exists(X)(X \neq Y \& c(X)) \Rightarrow o(Y)$	
7. $c(Y) \& \forall(X)(X \neq Y \rightarrow i(X)) \Rightarrow s(Y)$	

Table 1: Endorsements and Transformations for GET

where e_i and f_j are endorsements and the f 's comprise a subset of the e 's. A replacement rule is triggered when the endorsements comprising the left-hand side are in the knowledge base. The rule replaces the endorsements on the left-hand side with those on the right. The ability of replacement rules to update the knowledge base as information is acquired produces a non-monotonic support system. For example, the simultaneous presence of endorsements $c(X)$ and $i(X)$ causes the deletion of the consistency endorsement. The rules that define the propagation of support in GET are given in Table 1. The predicate $member(x, Y)$ is satisfied whenever x is a characteristic of Y .

A dominance relation on the derived endorsements is defined by rules 8–12. When a possibility X has been assigned endorsements specifying that it is both consistent and the sole consistent possibility, the former endorsement is removed since it is less informative than the endorsement $s(Y)$. Similarly, the presence of an inconsistency removes all endorsements designating consistency.

Initially, every possibility is assigned the consistency endorsement c . For a system to be consistent, it must agree with all of the acquired evidence. Two types of inconsistency can occur in a hierarchic reasoning system: *evidential* and *hierarchic*. Evidential inconsistency occurs when evidence is acquired that generates both $p(x)$ and $a(x)$. Hierarchic inconsistency results from the acquisition evidentially consistent information that is incompatible with each of the possibilities in the hierarchy.

Even when no possibility agrees with the totality of the evidence, the endorsements still contain information that may be used in determining a likely candidate. Evaluating

the likelihood of the alternatives requires the addition of a component that examines the composition of the knowledge base. In GET, the alternatives are ranked using the number of positive endorsements, negative endorsements and the number of characteristics that define the possibility.

For a possibility Y , let $pos(Y)$ denote the number of positive endorsements for Y . That is, the number of endorsements of the form $m(x, Y)$. Similarly, $neg(Y)$ denotes the number of negative endorsements. The support for a possibility is defined to be

$$sup(Y) = (pos(Y) - neg(Y)) / card(Y)$$

where $card(Y)$ is the cardinality of Y . A possibility X is deemed more likely than Y whenever X is consistent and Y is inconsistent or X and Y have the same consistency endorsement and $sup(X) > sup(Y)$.

The use of $card(Y)$ in computing $sup(Y)$ measures the lack of information concerning the characteristics of Y . Figure 2 traces the processing of information concerning the hierarchy in Figure 1. The possibilities are listed in the order specified by the ranking defined above. When evidence $p(a)$ and $a(d)$ is processed, $sup(P_1) = 1$ and $sup(P_3) = \frac{1}{3}$ even though both have one positive endorsement. P_3 is supported to a lesser degree since it contains elements for which no evidence has been acquired.

The final combination of evidence produces hierarchic inconsistency. The analysis designates P_3 as the most likely candidate since it has the most positive and fewest negative endorsements.

4 An identification system

Many identification problems acquire and evaluate information gathered from disparate sources. With this in mind, Borigda and Imielinski [1] proposed the process of decision making in a committee as a general framework for the analysis of uncertainty. In a committee deliberation, certain opinions carry more weight than others. This may be due to the level of expertise or the status (i.e. the chairman) of the committee member. An endorsement system can use multiple endorsements to determine a consensus of opinion. The strength of an endorsement may be reflected in combination rules and in the evaluation strategy.

A transformation based endorsement system was constructed to determine the identity of a person from descriptions of the physical characteristics of the person. Information for a database containing height, weight, sex and hair color was obtained, sometimes grudgingly, from the faculty and graduate students of the Wright State University computer science department. Evidence provided to the system consists of the quality of the observation and an estimate of a physical characteristic. An observation is either excellent and impaired; an impaired observation may be one made under less than ideal circumstances or by an inexperienced observer.

An observation generates endorsements for each person in the database. The endorsements indicate the proximity of the estimate to the recorded value. The endorsements are match (**ma**), possible match (**po**), unlikely (**un**) and improbable (**im**). The appropriate endorsement is determined by a range specified for each physical characteristic. For example, the weight endorsement is determined by the difference of the estimated weight (*wte*) and the weight recorded in the database (*wtp*) as follows:

ma	if $ wte - wtp \leq 5$
po	if $5 < wte - wtp \leq 10$
un	if $10 < wte - wtp \leq 15$
im	if $ wte - wtp > 15$

The endorsements for height are determined in a similar manner. A menu containing a spectrum of colors is given for the hair color estimate. A match endorsement is generated when the estimate is identical to the hair color in a database entry. The possible endorsement is generated if the estimate differs by only one position in the spectrum. For example, the possible endorsement is assigned to every person in the database whose hair color is brown or blond when light brown hair is specified by the observer. Match and improbable are the only endorsements assigned for the sex characteristic.

An endorsement has four arguments; the name of the person to whom the endorsement refers, the physical characteristic, a time tag and the quality of observation. The time tag is an integer that records the number of the observation that produced the endorsement. The endorsement

$po(johnsmith, weight, 5, ex)$

is generated when the database entry for John Smith's weight is 190 pounds and the fifth observation estimates the weight of the unknown individual as 183 pounds. Another observation that estimates the weight at 181 pounds produces an endorsement that differs from the preceding endorsement only in the time tag.

The cycle of evidence acquisition and endorsement generation follows the pattern presented in the previous section. The analysis of the endorsements establishes a measure of agreement between the observed physical characteristics and each person in the database. For each person *p* and characteristic *c*, the value $0 < agree(p, c) < 10$ is determined by the number and quality of the endorsements referring to that characteristic. Endorsements are assigned weights as follows:

	excellent	impaired
ma	10	6
po	7	2
im	3	0
un	0	0

To rank of the alternatives, we let $ex(p, c)$ and $im(p, c)$ denote the mean of the weights of the excellent and impaired endorsements for a person *p* and characteristic *c*, respectively. When there are no excellent observations, the evaluation uses the only information available. The acquisition of excellent observations reduces the dependence of the identification on less reliable information. This is reflected by degrading the significance of impaired observations.

excellent observations	$agree(p, c)$
0	$im(p, c)$
1	$\frac{7}{10}ex(p, c) + \frac{1}{2}im(p, c)$
2	$\frac{8}{10}ex(p, c) + \frac{1}{3}im(p, c)$
3	$\frac{9}{10}ex(p, c) + \frac{1}{6}im(p, c)$
4 or more	$ex(p, c)$

When there are four or more excellent observations, the impaired observations are no longer used. To obtain the highest possible rating, there must be at least two observations, one of which is excellent. Moreover, all of the observations must generate the **ma** endorsement. An individual's ranking is the sum of the values of the associated with the four characteristics.

The analysis of the alternatives in the identification system illustrates one of the fundamental properties of endorsement based reasoning. The value of an endorsement is dynamic, it may change as additional information is obtained and added to the knowledge base. The evaluation uses the information recorded in the endorsements to determine the weight of the evidence. This is what Sulli-

van and Cohen [6] refer to as explicitly reasoning with the causes of uncertainty rather than implicitly manipulating uncertainty through a numerical calculus. The endorsement system permits the re-evaluation of the significance of evidence based on the totality of all evidence that has been processed.

5 Conclusions

The systems described in this paper demonstrate a transformation based approach to the representation and combination of evidential support. Rules for the combination and propagation support are designed for the particular problem domain. The specification of knowledge base transformations as generation and replacement rules permits a straightforward translation of the system design into a Prolog implementation.

Advantages of endorsement based systems include the expressibility of the evidential representation and the flexibility of support propagation and evaluation techniques. Increasing the information in an endorsement provides additional capabilities to a symbolic reasoning system. Predicates can be added to replacement rules to produce time dependent analysis. The comparison of tags in endorsements e and f

$$e(i, Y), f(j, Y), i > j \rightarrow e(i, Y)$$

$$e(i, Y), f(j, Y), j > i \rightarrow f(i, Y)$$

defines a *recency* precedence of endorsements. In a time dependent problem domain, the dynamic capabilities of endorsement analysis can be used to give additional credence to recently obtained information.

In a symbolic reasoning system, the evidence and combining rules can be direct translations of domain information and reasoning. The endorsements in the knowledge represent the accumulated information. Unlike the numeric systems in which the alternatives are ranked by the associated values, assigning a measure of likelihood to the possibilities in an endorsement system is obtained by analyzing the contents of the knowledge base. Advances in endorsement based reasoning requires developing efficient techniques for evaluating the knowledge base.

References

- [1] A. Borigda and T. Imielinski. Decision making in committees: a framework for dealing with inconsistency and non-monotonicity. In *Non-Monotonic Reasoning Workshop*, pages 21–31, American Association for Artificial Intelligence, New Paltz, N. Y., 1984.
- [2] B. G. Buchanan and E. H. Shortliffe. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, 1984.
- [3] P. R. Cohen. *Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach*. Pitman Press, Boston, 1983.
- [4] J. R. Quinlan. Inferno: a cautious approach to uncertain inference. *New Generation Computing*, 26:255–269, 1983.
- [5] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.
- [6] M. Sullivan and P. R. Cohen. An endorsement-based plan recognition system. In *Proceedings 9th IJCAI*, pages 473–479, 1985.
- [7] A. Tversky and D. Kahneman. Judgement under uncertainty: heuristics and biases. *Science*, 185:1124–1131, 1974.

Reducing Uncertainty by Using Explanatory Relationships

John R. Josephson, Ph.D., The Ohio State University

Abstract

Explanatory relationships can be used effectively to reduce the uncertainty that remains after diagnostic hypotheses have been scored using local matching.

1. Introduction

The problem that the mind must solve is not that of reasoning with uncertainty, but reasoning *DESPITE* uncertainty -- how to come to robust conclusions despite uncertain data, inconclusive inference procedures, and incomplete knowledge.

(-- B. Chandrasekaran)

Suppose that some black box **hypothesis source** delivers up a set of diagnostic hypotheses, each hypothesis given a confidence value on some scale. Suppose further that these confidence values can be taken to reflect "local match" or *prima facie* likelihood. That is, the confidence value associated with each hypothesis is a measure of its likelihood of being true, based only on consideration of the match between the hypothesis and the data with little or no consideration of interactions between potentially rival or otherwise related hypotheses. Thus we have a picture of a set of hypotheses where each has been somehow stimulated, evoked, and instantiated for the case, and at the current stage of processing each hypothesis has been scored in isolation from the others.

At this stage we have both a problem, and an opportunity to do something about it. The problem is that many hypotheses will probably have intermediate scores, representing hypotheses that can neither be taken as practically certain, nor as being of such a low confidence as to be ignorable. Some number of these hypotheses are presumably true, but how many and which ones?

The opportunity is that of bringing knowledge of interactions between the hypotheses to bear in order to reduce the degree of uncertainty associated with the hypotheses -- increasing confidence in some of them, and decreasing confidence in others.

Some types of interactions between hypotheses are:

- A and B are mutually incompatible.
- A is a more detailed refinement of B.
- A could be caused by B.
- A and B are mutually compatible, and are explanatory alternatives where their explanatory coverages overlap.

Besides hypothesis-hypothesis interactions of mutual incompatibility and support, which can arise by degrees as well as discretely, one especially interesting class of interactions concerns explanatory relations: what happens when two or more hypotheses represent alternative explanations for the same datum? The focus of this paper will be on explanatory relationships, and on how explanatory relationships impact on our estimates of confidence.

Knowledge of explanatory relationships gives us an opportunity to take advantage of some "best explanation" reasoning.

2. Best-Explanation Reasoning

Inference to the Best Explanation or *Abduction* is a form of inference that follows a pattern approximately like this:^{1, 2, 3, 4}

D is a collection of data (facts, observations, givens),
H explains D (would, if true, explain D),
No other hypothesis explains D as well as H does.

Therefore, H is correct.

The strength of an abductive conclusion will in general depend on several factors, including:

- how good H is by itself, independently of considering the alternatives,
- how decisively H surpasses the alternatives,
- how thorough the search was for alternative explanations, and
- pragmatic considerations, including
 - the costs of being wrong and the benefits of being right,
 - how strong the need is to come to a conclusion at all, especially considering the possibility of seeking further evidence before deciding.

Abductions, as we have just characterized them, go from data describing something to an explanatory hypothesis that best accounts for that data.

3. Using Explanatory Relationships

Let us suppose that, besides a confidence value, each plausible diagnostic hypothesis (one which is not ruled-out) is associated with a description of which findings that hypothesis can explain.

One way to take advantage of these explanatory relationships is to set up a standard for when a diagnosis is complete. The diagnosis can be considered to be complete when all of the abnormal findings have been accounted for (explained). (This standard should be considered to be somewhat of an idealization, since for example unimportant findings need not be accounted for.)

Let us focus on the use of explanatory relationships to reduce the uncertainty that remains after the confidence scoring based on local matching. First we note that an overall abduction problem is set up - to account for all of the (abnormal) findings, and a series of small abduction problems is set up - to account for each particular (abnormal) finding. Our basic strategy will be to try to solve the overall abduction problem by solving some number of smaller and easier abduction problems.

First we solve the easiest little abduction problems, the ones in which we can have the most confidence. If a certain hypothesis is the only plausible explanation for some finding, then (supposing its local-match confidence value is not too low) it is entitled to as high confidence value, and entitled to be accepted into the overall composite hypothesis that represents the solution to the overall abductive problem. So first we form the set of **Essential** hypotheses consisting of those of the sort we have just mentioned.

If we are lucky the set of Essential hypotheses will together account for all of the (important abnormal) findings. If this occurs then the overall abduction problem is solved - the set of Essentials together constitutes the best explanation - and the diagnosis is complete. Hypotheses which are not part of this best explanation are lowered in confidence, since they are not needed as part of the final explanation, and everything that they explain can now be explained in some other (and in the context better) way.

If the Essentials do not explain everything, then next we form the set of **Clear Best** hypotheses consisting of those which explain findings for which there is no other explanation anywhere near as good. For example if some finding S can only be explained by A (moderate confidence), B (low confidence), and C (low confidence), then A is worthy of acceptance as being clearly the best way to explain S. Hopefully, the Essentials together with the Clear Bests will now explain everything, and the diagnosis can be considered to be complete (after perhaps removing some few hypotheses which are now explanatorily superfluous in the presence of the rest).

If the Essentials together with the Clear Bests do not explain everything we have done all we can do on the current evidence without resorting to guessing. Generally our best strategy under these circumstances would be to gather more data. In fact we are

in a position to guide our data gathering by focusing on the problem of discriminating between alternative good explanations for important findings.

Yet sometime we have to decide quickly, and do not have enough time to gather further data. Also sometimes the cost of gathering further data is too high. Under these circumstances we still have the means available to do some clever guessing. We can begin to include hypotheses which are best explanations for certain finding, but which are not far enough ahead of the alternatives, or not of high enough local-match confidence, to enable them to be accepted confidently. These Weakly Bests constitute the best guesses we can make under the circumstances.

Actually we can even do slightly better. Findings can be made to *vote* for the hypotheses which best explain them. The idea is that two different findings both pointing to the same hypotheses as the best explanation constitute (apparently) independent sources of evidence for the hypothesis, i.e. constitute converging lines of inference for the hypothesis. Hypotheses with more votes can be accepted more confidently than hypotheses with fewer votes, and enough can be accepted to complete the explanation. This phenomenon of converging lines of inference seems to be what the philosopher of science William Whewell (1794-1866) called "the concilience of inductions."

4. Conclusion

We have shown how a stage of diagnostic problem solving, where there are N viable plausible hypotheses, each with a confidence score based on local matching, can, by using explanatory relationships, be brought to a more advanced stage, where the number of hypotheses has been radically pruned to k hypotheses together representing a single compound hypothesis that explains a distinct portion of the data, and which is a "logically optimal" outcome in an abductive sense. I should point out that variations of this method have played an important role in several knowledge-based systems including the Red system for Red-cell antibody identification⁵ and the Pathex system for diagnosing cholestatic liver diseases.

Note that, at the level of description we have been using, we might be describing the information processing of an "algorithmic computer", i.e. an instruction follower; or a "connectionist computer", i.e. one whose primitive processing elements work by propagating nudges and activation strengths. In either case what we are describing is the functional and semantic significance of various actions of the machine, not precisely how these actions are accomplished.

Note too that we might be describing a medical diagnosis engine, or a diagnoser of mechanical systems, a fragment of the processing of the vertebrate visual system, or the information processing that goes on when we recognize words in continuous speech. The strategy for reducing uncertainty that we have described is appropriate quite generally for a variety of abductive or interpretive information processing tasks.

5. Acknowledgments

Michael C. Tanner and Ashok Goel of Ohio State have contributed significantly to the development of these ideas, which have also benefited greatly from discussions with Jack Smith, Jr. and William Punch. This research has been supported in part by the Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010 under the Strategic Computing Program, in part by the National Library of Medicine under grant LM-04298, and in part by the National Heart Lung and Blood Institute under grant HL-38776. Computer facilities have been enhanced through gifts from Xerox Corporation.

References

1. Josephson, John R., "A Mechanism for Forming Compositer Explanatory Hypotheses", *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Causal and Strategic Aspects of Diagnostic Reasoning*, Vol. SMC-17, No. 3, May, June 1987, pp. 445-54.
2. Peirce, C.S., *Abduction and Induction*, Dover, 1955, pp. 150ff., ch. 11.
3. Pople, H., "On the Mechanization of Abductive Logic", *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 1973, pp. 147-152.
4. Eugene Charniak and Drew McDermott, *Introduction to Artificial Intelligence*, Addison Wesley, 1985.
5. Smith, Jack W.; Svrbely, John R.; Evans, Charles A.; Strohm, Pat; Josephson, John R.; Tanner, Mike, "RED: A Red-Cell Antibody Identification Expert Module", *Journal of Medical Systems*, Vol. 9, No. 3, 1985, pp. 121-138.

SPACEBORNE VHSIC MULTIPROCESSOR SYSTEM
FOR AI APPLICATIONS

Henry Lum, Jr.
NASA Ames Research Center
Moffitt Field, CA 94040

Howard E. Shrobe* and John G. Aspinall
Symbolics Cambridge Research Center
11 Cambridge Center
Cambridge, MA 02142

Abstract

A multiprocessor system, under design for space-station applications, makes use of the latest generation symbolic processor and packaging technology. The result will be a compact, space-qualified system two to three orders of magnitude more powerful than present-day symbolic processing systems.

1 Symbolic Computing

The tasks for which symbolic computing is uniquely qualified are different from those served well by conventional numerical computing. Conventional programs tend to be uniform, simple, homogeneous, and numerically intensive. Symbolic programs, on the other hand, are diverse and heterogeneous, involving a variety of mechanisms and conceptual tasks within a single program. A single symbolic computing application, for example the management of an autonomous space vehicle, will have to perform a variety of tasks such as hierarchical classification, signal interpretation, hypothesis formation, matching, and logical inference; not to mention conventional numerical tasks. It will have to employ a variety of different mechanisms such as rule-based programming, frame-instantiation, constraint propagation, numerical simulation, object-oriented programming, symbolic mathematics, and truth maintenance; all within a single large system.

The popular notion of an AI program as a single, simple rule interpreter is a gross oversimplification. In fact, symbolic computing places much more serious demands on the system architecture than would be presented by the need simply to support a simple rule interpreter.

1.1 The Object-Oriented Viewpoint

A viewpoint of a computer that is characteristic of symbolic computation is called the *Object-Oriented*

*Howard Shrobe is also a Principal Research Scientist at the MIT Artificial Intelligence Laboratory.

Viewpoint. In this viewpoint memory does not consist of a stream of raw bits organized into bytes or words. Rather, it consists of much larger conceptual entities which are thought of as objects. An object might be something simple like a list, an array, an integer or it might be something with higher semantic content, for example, a node in a semantic network or a data structure representing an entity in the real world.

These objects should have an identity. This means that you should be able to tell the type of an object, just by looking at it. In addition, one should be able to tell its location in memory. The techniques that are used to do this are called *storage conventions*. Ideally, the hardware should guarantee that the storage conventions are never violated.

The object-oriented viewpoint depends upon the ability to make memory seemingly infinite, in the sense that there will always be room for allocating new objects. Indeed, the goal is to free the programmer from worrying about where objects are allocated and when they are deallocated. In practice, this means that the system needs to support garbage collection, the process of reclaiming unused storage. It is necessary that unused storage be reclaimed at a rapid enough rate so that free storage is always available. Garbage collection means that the symmetry of storage is maintained; to the programmer, all storage is the same and its always available.

The second major feature of the object-oriented viewpoint is that the programmer codes using *Generic Operations*. A generic operation is defined as an abstract, conceptual operation which does not reflect the limitations of the hardware. For example, addition is a conceptual operation which is meaningful to apply to integers, floating-point numbers, vectors, polynomials, etc. Ideally, there should be a single operation, called PLUS, which does all of these, dispatching on the type of the objects being added to determine how to perform the operation.

Modern symbolic computing hardware allows this viewpoint to be supported efficiently. It is the hardware's job to check every operation and decide how to perform it based upon the types of the operands. So in effect that hardware will tell itself: "That's a

PRECEDING PAGE BLANK NOT FILMED

fixed-point number and therefore I should do integer add," or, "That's a floating point number, I should be performing floating point add." Or, "It's an extended number that I can't directly support at all, but I can support it by this sequence of other instructions."

In addition to higher level code, this approach leads to better debugging capability and supports the concept of incrementality. Since dispatching on the operand type is a runtime function, a new data-type may be added by simply defining the generic operation upon the new type. Existing software can now use the new data-type without recompilation. Any attempt to do an invalid operation on any particular piece of data is detected by the hardware, allowing the programmer to enter a debugging session in the context of the error.

2 Ivory

Symbolics is now implementing a new generation of symbolic processing architectures built upon the Ivory processor. Ivory-based architectures represent the state of the art in satisfying the requirements of symbolic processing, as described in the previous section. In particular, Ivory supplies the following.

- Runtime type checking - Parallel tag processor, late-branch ROM and comprehensive trap logic support generic arithmetic and pointer manipulation.
- Virtual Memory Support - On chip translation buffer, microcoded cache-miss backup and the support of CDR-coded lists (more compact physical memory representation).
- Specialized Lisp operations - Pipelined memory interface and high level microcoded primitives support efficient implementation of operations such as CAR and CDR.
- Garbage Collection - On chip hardware to facilitate efficient GC algorithms such as the Ephemeral GC [Moon, 1984].
- Fast call and return - Specialized datapaths, parallel operations, and fast cycle time support the complex calling strategies required by Lisp.
- Fast "vector" instructions for garbage collection, data-base searching and graphics applications.
- A fast coprocessor interface, primarily used to provide high floating point performance.
- A programmable interleaved memory interface to allow a wide range of memory system speeds and architectures to be used - ranging from small high speed caches to four-way interleaved standard MOS memories.

2.1 Data Architecture

In line with the requirements of the object-oriented viewpoint, every word in memory contains either an

object reference or is part of the representation of an object. A machine word contains 40 bits, which are assigned as in Figure 1.

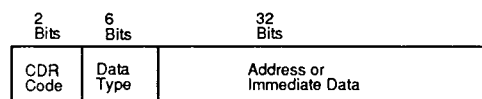


Figure 1: Ivory Memory Word

The *data type field* indicates what kind of information is stored in a word. The *cdr-code field* is used for various purposes. For header data types, the cdr-code field is used as an extension of the data-type field. For stored representations of lists, the contents of this field indicate how the data that constitute the list are stored. This results in a compact representation of lists. The *address or immediate data field* is interpreted according to the data type of the word. This field contains either the address of the stored representation of an object, or the actual representation of an object.

Ivory supports the rich variety of objects found in symbolic processing environments as described in the previous section. General Lisp data structures such as symbols, lists, arrays, strings, and characters are all directly manipulated by the instruction set. For numeric data types, Ivory includes very efficient support (immediate object representation) for 32-bit integers and 32-bit IEEE single-precision floating point numbers. It also supports infinite precision integers, 64-bit IEEE double-precision numbers, rational numbers, and complex numbers.

2.2 Virtual Memory

Ivory implements a 4 gigaword virtual address space. The 32-bit virtual word address is divided into a 24-bit virtual page number and an 8-bit page offset. The virtual page number is mapped via the *Page Hash Table* (PHT) to get a 24-bit physical page number.

While the 256-word page size may seem small by traditional processor standards, it is appropriate for symbolic processors. Symbolic processors tend to have many small functions, small data objects, and little locality of reference. These factors tend to limit the advantages of a larger page size, and the smaller page size allows better allocation of physical memory.

2.3 Garbage Collection

The Ivory memory architecture supports two methods for garbage collection (GC). Both strategies are incremental in nature and identical to the Symbolics 3600 implementation [Moon, 1984; Moon, 1985]. The two methods differ in how they decide to actually reclaim storage. In both cases the garbage collection

process *condemns* or identifies storage it would like to reclaim. This storage is considered to occupy *old space* while other storage is termed *new space*. If the processor attempts to read an object reference to old space, a trap will be taken and the *Transporter* will be invoked. This is a software routine which copies the storage containing the object representation into new space. It also updates the pointer to the old object in memory to point to the copy of the object in new space. In order to signal the trap which invokes the transporter the memory interface looks at the data type of a word to determine if it is a pointer, and if the address field points to old space.

The *Dynamic GC* is used to reclaim objects that have lifetimes on the order of tens of minutes to hours and days. It performs a single linear scan of all of new space, reading every memory word. During this scan the memory interface will cause the transporter to be invoked if the word is a pointer to old space. At the end of the scan, all of new space must point only to new space and the storage used by old space can be reclaimed. This scan is done incrementally so as not to hurt interactive performance.

The *Ephemeral GC* (EGC) is used for reclaiming objects that have lifetimes of the order of seconds to minutes. This scheme is based on the observation that most of the objects created in the system have a relatively short lifetime. The EGC attempts to reclaim the most recently allocated objects by breaking up storage into *levels*, corresponding to how recently an object was created. Ephemeral GC requires the memory system to maintain a database of pages which contain pointers to a more recent level. When the EGC condemns the most recent level it uses the database to scan only those pages which potentially contain pointers to the condemned level. To support this, the memory interface must notice when it is writing a pointer to a more recent level into a page. In Ivory, this information is maintained in the PHT for pages which are in physical memory, and in a companion structure for pages which reside in secondary storage.

2.4 Stack Execution

Ivory uses a stack-based model of execution. The stack is divided into frames, one for each active function. The stack is used for passing arguments, allocation of local variables, and intermediate results of computations. A stack frame is indexed by three pointer registers; the frame pointer (FP), the local pointer (LP), and the stack pointer (SP).

The frame information consists of two words; the offsets of the LP and SP registers from the FP, and the continuation of this frame. The continuation is either the return address of this function, or the next function to call. The FP is used to access the frame information and the arguments to the function. The SP is used to access intermediate results of computations. The LP must be distinct from the FP because arguments are pushed by the caller, and may be pushed in

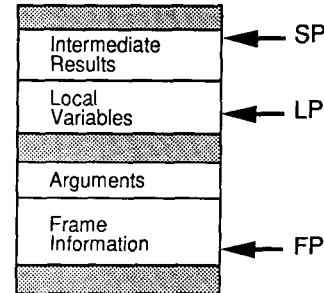


Figure 2: Ivory Stack Frame

several different ways.

2.5 Instruction Set

Ivory performs different operations depending on the data-type of the word that is fetched as an instruction. Most object references push themselves onto the top of the stack. This capability is used to supply full word constant operands. A special data-type is used to push the contents of the word whose address is specified in the address field of the instruction word. Other data-types are allocated to perform specialized types of function calls. There are data-types for calling compiled functions and generic functions using the address field to point to the function. A further type is used to call the contents of a memory word as a function. This is used to implement dynamically linked functions. Finally there is a set of 16 data-types which are further decoded into two "packed" instructions. The 32-bit immediate data, along with 4 bits from the data-type field are combined to form two 18-bit instructions.

The Ivory instruction set incorporates full run-time error and exception checking. Exceptions are cases which are not an error, but cannot be handled by the processor hardware without software intervention. The checking performed by the instruction set includes full checking of data-types, subscript range, uninitialized variables, undefined functions, and detection of integer and floating-point overflow. Exceptions are handled by causing a trap through a vector in memory.

Even though strict error-checking performed is by the instruction set, it is possible to extend the instructions supplied to handle new object types. By utilizing exception handlers and the New Flavors [Moon, 1986] object-oriented programming system, it is possible to define a new object type that masquerades as an existing type in all programs. Conversely it is possible to treat the architecturally-defined data-types as part of the object-oriented system. This permits the def-

	I	D	E	C
	ADD	PUSH B	PUSH A	-
PROGRAM:	POP C	ADD	PUSH B	PUSH A
PUSH A	-	POP C	ADD	PUSH B
PUSH B	-	-	POP C	ADD
ADD	-	-	-	POP C
POP C	-	-	-	-

Figure 3: Ivory Pipeline Stages

initiation of generic functions (as described in the first section of this paper) which can operate uniformly on instances and objects with different data-types.

2.6 Microarchitecture

The Ivory microprocessor implements a 4-stage pipeline as shown in Figure 3. The first stage fetches the instruction, decodes it, and adjusts the program counter. The second stage fetches the initial microinstruction, computes the operand address, and adjusts the stack pointer. The third stage fetches the operands and computes the result. The fourth stage stores the result, unless a fault has occurred, in which case it restores the state of the third stage.

Instructions spend only a single cycle in the first two pipeline stages, but can spend an arbitrary number of cycles in the execute stage. Simple instructions, such as PUSH, ADD, and EQ execute in a single cycle. Conditional branches are resolved in the second (D) stage. A taken conditional branch executes in two cycles; a not-taken branch in one.

Figure 4 shows a block diagram of the Ivory CPU. Instructions are fetched directly from a 32 word (up to 64 instructions) direct-mapped instruction cache. The cache, which is filled by an autonomous prefetcher, serves to buffer instructions arriving from memory and hold small program loops. A bypass path provides the instruction directly from memory when the first stage is stalled on a cache miss.

The operand address calculation data path contains the stack frame pointer registers and a 32-bit adder/subtractor. It computes the address of the operand and stack pointer adjustment according to the macroinstruction. This data path is also used in parallel with the main data path to accelerate function call/return.

The main data path contains a 128 word top-of-stack cache, a 32-word scratchpad (which contains a duplicate of the top word on the stack in a fixed location), the ALU, and tag checking logic. The ALU includes an adder, boolean unit, shift/mask logic, and support for one-bit-per-cycle integer multiply/divide.

Tag checking is done in parallel with the ALU operation, so that in the common cases no time penalty is paid for type checking. Similarly, ECC checking of data from memory is done in parallel with the ALU using the on-chip ECC logic. Bypass paths for both the top-of-stack cache and scratchpad forward the result of the previous instruction to the ALU as necessary.

The Ivory processor supports a pipelined memory bus which can have up to four outstanding requests at once. An associative queue of outstanding request addresses is maintained for detecting when instructions arrive from memory and installing them into the instruction cache. The memory interface protocol is implemented by an independent state machine which arbitrates between on-chip users of the memory system and other bus masters.

3 Multiprocessing with Ivory

In addition to the features of Ivory described in the previous section, there are several design features of the Ivory processor specifically intended to support multiprocessor architectures. They are:

- Support for Futures.
- Support for Special Variable Binding.
- Synchronization primitives.

3.1 Futures

Futures are a Lisp language construct which appear in parallel extensions to Lisp such as MultiLisp [Halstead, 1985] and QLISP [Gabriel & McCarthy, 1984]. A future is a compound structure which represents a promised value coupled with a process that computes the value. The future is a first class data structure which can be stored in other data structures, loaded and stored even though the process computing its value has not terminated. However, if the value of the future is ever required for a computation (e.g. it is one input to an arithmetic operation) then the processes attempting to *touch* the future blocks until the future's value is delivered. This facilitates a very flexible, demand driven style of parallel processing.

Ivory provides a special hardware datatype for futures which is known about by the microcode and the tags processor. This datatype acts as an invisible pointer; if a future has been delivered, the instruction attempting to use its value is not interrupted, but simply follow the future pointer to its actual value. If the value has not been delivered, the hardware causes a trap; the operating system can then suspend the requesting process until the value is delivered.

Without the hardware support provided for the future datatype, the compiler would have to emit code to check the datatype of *every* value manipulated by the program. This is because any value might be a future. Experiments with the QLISP system at Stanford University have shown that this leads to unac-

ORIGINAL PAGE IS
OF POOR QUALITY

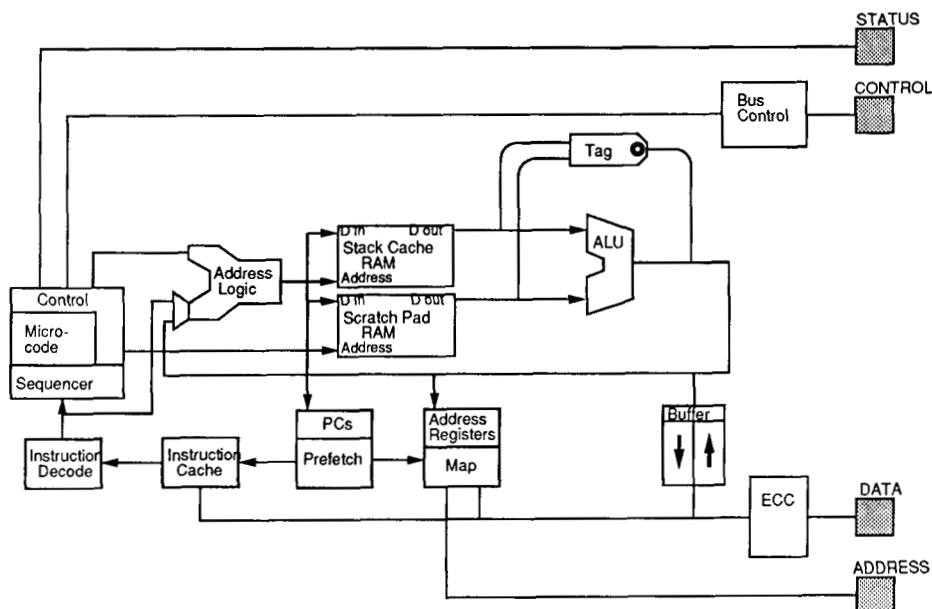


Figure 4: Ivory Processor Block Diagram

ceptable overhead in their implementation on stock parallel processing hardware.

3.2 Special Variable Bindings

Lisp allows two types of variable bindings. Lexical binding of variables is the type familiar in all block structured languages. Dynamic binding (also known as special binding), however, changes the globally accessible value of a location through the dynamic extent of the binding. This change, however, is visible only within the process which bound the variable; all other processes see either the global value or their own dynamically bound value.

Classically, special variable binding is performed using shallow binding. This involves overwriting the location with its newly bound value while saving the old value in a special stack. Shallow binding optimizes the speed of access to the variable. Shallow binding can be (and is) used in sequential machines that support multiple processes. When a process relinquishes control of the processor, its special variable bindings are undone; the process which gains control of the processor must first establish its special variable bindings before beginning its execution. This makes processes switching costly, even for sequential machines.

In the parallel processing world the shallow binding technique doesn't work at all. This is because two distinct processors can be concurrently executing separate processes each of which wants to bind the location to a unique value. Since shallow binding works by overwriting the single location there is no way for two processes both to bind the same location.

This forces the use of the much slower deep binding technique for special variables. In a deep binding scheme, each process maintains an ordered mapping between locations and bound values. The mapping must be ordered since a process can repeatedly rebind the value of a single location and the latest binding should hold. This data structure must be sequentially searched for a variable binding; this is typically a very slow process.

Ivory provides hardware support to optimize deep binding. A special datatype, called *bound location*, is used to indicate that a location has been dynamically bound. Whenever Ivory encounters such a datatype, it traps to a microcode routine that searches a hashtable for the value of the binding. The hashtable uses a key derived from both the identity of the binding process and the address of the location bound. The binding and unbinding instructions keep this hashtable up to date. A probe into this table is very fast;

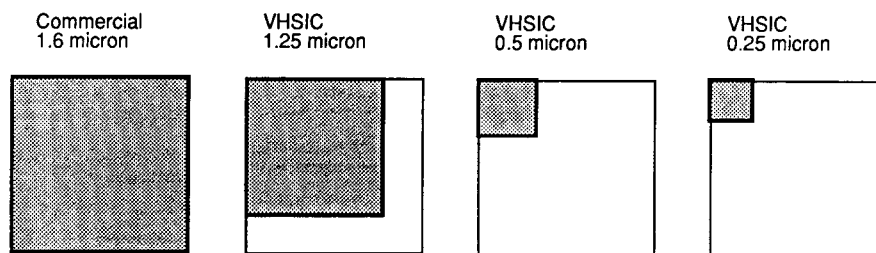


Figure 5: Relative sizes of a single Ivory processor in four different technologies.

if there is no entry in the hash table, then a classic deep binding search is initiated.

This technique has two advantages over the techniques possible without datatype checking hardware. First, for locations which have not been dynamically bound, there is no cost above that of shallow binding, since the special techniques are only invoked for locations whose datatype is *bound location*; these markers are only placed in a location that is dynamically bound. The second advantage comes from the hardware assisted hashing used to fetch the binding. In conventional processors neither of these techniques are available. The lack of the bound location datatype is particularly critical since any location may be dynamically bound and hence any load or store must check for this.

3.3 Locking and Synchronization

Parallel processing in the presence of side-effects requires techniques for establishing critical regions, mutual exclusion from data structures and joint synchronization of processors to rendezvous points. These are very difficult to achieve efficiently without hardware support. Ivory provides a *store conditional* instruction that can serve as the basis for all of these facilities. Store conditional takes three arguments; the first is a location, the second two are the new value and the test value. If the location currently contains a value EQ to the test value, then the new value is stored in the location. The value returned by the instruction can be tested to see if the store succeeded.

Semaphores, atomic updates and locks can all be implemented using this single atomic update primitive.

4 Technology for a Spaceborne Processor Architecture

The ultimate Spaceborne VHSIC multiprocessor will result from a combination of the powerful computer architecture ideas of Ivory with evolving VHSIC hardware technologies. At each stage of this evolution, the overall performance, integration and reliability of the

system will be increased. Three hardware technologies are particularly important:

- Fine-line VHSIC chip technology, moving from 1.25 micron, to .5 micron and finally to .25 micron Rad-Hard CMOS.
- Super-chip technology which integrates a significant portion of the total system onto a single large die (2 inch square), using redundancy techniques to achieve adequate yields.
- Button-Board System Packaging which allows very dense packaging of boards into modules without the use of backplanes and connectors.

4.1 Chips and Superchips

The first commercial Ivory chip is implemented in 1.6 micron CMOS technology and runs with a cycle time in the vicinity of 150 ns. At this clock rate Ivory's performance is roughly 3 times that of the Symbolics 3600. A second version of Ivory is now available with cycle times in the vicinity of 100 ns.

Radiation doses as high as 10^5 rads are expected in the space station environment. Since mechanical shielding takes up space and weight, the use of rad-hardened technology is preferable. VHSIC provides a CMOS technology with adequate radiation resistance for space-station applications. In addition, error-correcting logic on the memory bus of Ivory (already provided in the commercial version of Ivory) may be enhanced for increased reliability to single event upset (SEU).

Figure 5 shows how the die size for an Ivory chip will shrink as it is reimplemented in newer VHSIC technologies. Since Ivory is implemented in a technology independent design system, it can be retargeted to these new technologies in a matter of days through a totally mechanical process. These dense processes also allow the cycle time to be reduced; cycle times below 50 ns should be achievable with the .5 micron process.

Figure 6 shows the outline of a super-chip containing Ivory chips implemented in .5 and .25 micron technology. Even with several redundant copies of the

ORIGINAL PAGE IS
OF POOR QUALITY

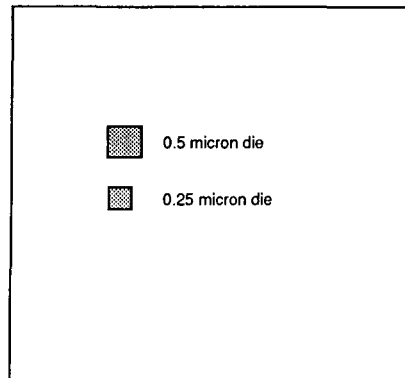


Figure 6: Ivory processors in a superchip.

Ivory die, the superchip contains room for an extensive cache memory. The advantage of this approach is that the cache can be more closely coupled to the processor, avoiding the delays associated with crossing chip boundaries.

Using this technology also allows the chip architecture to evolve further, leading to additional performance improvements. Such processors should be capable of performance in the range of 15 to 20 times that of the Symbolics 3600.

For small to modest scale parallelism, two interconnection technologies are appropriate. For system of 4 to 8 Ivory processors, a shared bus with snooping caches is capable of providing adequate bandwidth and a coherent memory image shared by all processors. However, a single bus system is not attractive when fault-tolerance considerations are added in.

A cross-bar interconnection scheme can support a larger number of processor (up to 32) and provide greater fault tolerance. Figure 7 shows an example of a crossbar interconnect.

4.2 Button Board Interconnect

Buttonboard Packaging is a new technology at TRW which provides performance and density improvements over those possible with conventional packages. Buttonboard packaging replaces bulky edge connectors with "button"-shaped contacts and small PC-board strips with low interconnect density and few layers. Buttons may be placed anywhere on a circuit board to provide an interconnect between boards. This allows a reduction of the path length of inter-board signals, thereby reducing propagation delays.

In the envisaged design, component-carrying boards alternate with signal-routing boards in a stack which

is fastened together to make a monolithic whole. Results of prototype testing seem to indicate that a buttonboard-based design will easily meet electrical and mechanical requirements for the space station.

Figure 8 shows how a crossbar system can be implemented using button board packaging. One interesting feature of this packaging technique is that additional processors (or memory) can be added simply by dropping in an additional processor card. A full scale SVMS system should therefore capable of a peak performance of greater than 500 times that of the Symbolics 3600.

References

- [Baker *et al.*, 1987] C. Baker, D. Chan, J. Cherry, A. Corry, G. Efland, B. Edwards, M. Matson, H. Minsky, E. Nestler, K. Reti, D. Sarrazin, C. Sommer, D. Tan and N. Weste. The Symbolics Ivory Processor: A 40 Bit Tagged Architecture Lisp Microprocessor. In *Proceedings ICCD-87*.
- [Edwards *et al.*, 1987] B. Edwards, G. Efland and N. Weste. The Symbolics I-Machine Architecture: A Symbolic Processor Architecture for VLSI Implementation. In *Proceedings ICCD-87*.
- [Gabriel & McCarthy, 1984] R.P. Gabriel and J. McCarthy. Queue-based multiprocessing Lisp. Symposium on Lisp and Functional Programming, August 1984.
- [Halstead, 1985] Halstead, R. MultiLisp: A Language for Concurrent Symbolic Computation ACM TOPLAS, October 1985.
- [Moon, 1984] D.A. Moon. Garbage Collection in a Large Lisp System. 1984 ACM Symposium on Lisp and Functional Programming, August 1984, pp. 235-246.
- [Moon, 1985] D.A. Moon. Architecture of the Symbolics 3600. Proceedings of the 12th IEEE International Symposium on Computer Architecture, 1985, pp. 76-83.
- [Moon, 1986] D.A. Moon. Object-Oriented Programming with Flavors. Proceedings of OOPSLA '86, pp. 1-8.
- [Shrobe *et al.*, 1987] H.E. Shrobe, J.G. Aspinall and N.L. Mayle. Towards a Virtual Parallel Inference Engine. To appear in *Proceedings AAAI-88*.

**AUTONOMOUS SATELLITE COMMAND AND CONTROL:
A COMPARISON WITH OTHER MILITARY SYSTEMS**

Robert J. Kruchten and Wayne Todd, Capt, USAF
Rome Air Development Center/COAD
Griffiss Air Force Base, New York 13441-5700

ABSTRACT

Existing satellite concepts of operation depend on readily available experts and are extremely manpower intensive. Areas of expertise required include mission planning, mission data interpretation, telemetry monitoring, and anomaly resolution. The concepts of operation have evolved to their current state in part because space systems have tended to be treated more as research and development assets rather than as operational assets. These methods of satellite command and control will be inadequate in the future because of the availability, survivability, and capability of human experts. Because space systems have extremely high reliability and limited access, they offer challenges not found in other military systems. Thus, automation techniques used elsewhere are not necessarily applicable to space systems. RADC has developed a program to make satellites much more autonomous using a variety of advanced software techniques. The purpose of this paper is to present the problem the program is addressing, some possible solutions, the goals of the RADC program, the rationale as to why the goals are reasonable, and the current program status. Also presented are some of the concepts used in the program and how they differ from more traditional approaches.

1. Introduction

Operation and control of satellites can be divided into two major areas: health/status and mission. Health/status is a broad definition covering all activities not directly concerned with executing the primary mission of the satellite. This includes power control, thermal control, attitude control, telemetry collection/formatting, station keeping, overall monitoring to insure nominal operation, and anomaly resolution. Anomaly resolution consists of detecting and diagnosing a real or apparent anomaly(ies), developing recommended courses of action to resolve the anomaly(ies), executing one or more courses of action, and observing the results of that action(s). These health/status activities are somewhat generic between different satellites although the details will differ.

The mission activities may involve situation assessment, scheduling, tracking, processing preplanned and real-time user requests, and mission data interpretation. These mission activities are not as generic as health/status activities because they are more dependent on the type of satellite. For example, mission activities of a communications satellite may be significantly different than a surveillance satellite or a navigation satellite.

Both health/status and mission are potential targets for greater autonomy. Each relies on ground support from experts with highly specialized knowledge. The basic goal of satellite autonomy is to transfer this knowledge to the satellite and employ it so the satellite achieves much greater independence from the experts and the ground support infrastructure they require. The result will be a command and control system that is both affordable and survivable for future operations.

2. Problem

Current methods of commanding and controlling both the health/status and mission activities have limitations that must be addressed to insure effective satellite operations in the future. The problem areas considered herein are cost, time delays, survivability, the increasing dependence on experts, and some unique aspects of this problem compared to other systems.

2.1. Cost

The command and control cost of satellite assets is measured in terms of the resources consumed, including manpower and facilities. Current operations are costly and the cost will likely become prohibitive in the future without significant changes in operational concepts.

As the number of space systems grows, it will become increasingly difficult, if not impossible, to find enough qualified personnel to operate and control these systems. This is especially true in the case of the technical experts who evaluate situations that are unusual, complex, and difficult to diagnose. These experts must also recommend actions; such actions may be novel

solutions for unanticipated problems. A significant portion of the dollars expended in operating and maintaining space systems is for personnel. Again, the key problem lies with those experts used to deal with difficult situations. Thus, even if enough people could be found to perform the expert functions, the money required may be prohibitive. This could have a major impact on an effort such as the Strategic Defense Initiative (SDI) which may involve many more satellites than are currently supported. One way to reduce to number of experts required for satellite support is to centralize command and control facilities. However, this can also create more time delays and reduce survivability as described below. Also, centralized command and control doesn't ameliorate the need for numerous geographically-dispersed transmit/receive ground stations which experts need to maintain frequent satellite contact during difficult situations.

2.2. Time Delays

One of the concerns with centralized command and control is that it can cause delays by requiring individual users to communicate directly or indirectly with the central control. This becomes necessary because the ground control must resolve conflicts in user requests and insure the system constraints are always satisfied. Even ignoring the delays entailed by requiring central coordination of user requests, the current methods of satellite control are not particularly fast. This is especially true when unusual situations arise. That is, situations that are not covered by routine operating procedures require extensive expert analysis before resolution. Unique satellite failures are one example of an unusual situation. Many satellite failures take weeks or, in some cases, even months to totally resolve. During this time, the satellite may be safed and often cannot fully meet its mission goals. As new requirements come into play (such as SDI), response time will become more critical. Not only will it be necessary to take an immediate response to reach a safe state, but it will also be necessary to quickly reach a final decision on a situation. Fortunately, efforts to improve response time can also increase efficiency. By speeding the decision process, resources on-board the satellite can be used more effectively since more windows of opportunity are available. The improvement could be faster recovery from an anomalous condition or it may even be redirecting mission functions faster than ground controllers are able.

2.3. Survivability

Most of today's space systems utilize a centralized command and control concept for health/status and mission functions. Although this can reduce cost, it can also create survivability problems. That is, a centralized command and control facility in a fixed location is easier to target (destroy or electronically jam) than several mobile targets. Multi-node, mobile, distributed control systems for health/status have been suggested to improve survivability because such systems would be

difficult to target or jam. Unfortunately, it is not possible to simply relocate all personnel controlling today's satellites into mobile systems. Many people would simply be unwilling to work in a mobile (especially remote) facility. Also, the savings in personnel and facility costs, gained through centralized control, would be lost.

2.4. Increasing Expert Dependence

Current space systems are controlled by technical personnel assisted by experts (usually from the manufacturer of the satellite) who evaluate complex situations and recommend actions. These are highly technical people with long experience in satellite design and operation. Interestingly, some efforts to introduce more automation into satellites may actually compound the need for these experts. This is because, although the automation reduces the efforts required by technicians, it can greatly increase the complexity of the satellite system in both the health/status and mission areas. This, in turn, can create a higher dependence on experts for difficult situations or problems.

Another problem is beginning to occur in space systems with long life spans, a characteristic which is generally desirable for an expensive, inaccessible asset. The problem is that a satellite can "live" long enough for the original experts to retire or move on to other programs. Incoming people are not as familiar with the satellite's design or history (i.e., heuristics of operation). This creates a greater need to capture this knowledge before the experts leave the program. Some NASA programs are recognizing this problem and attempt to create historical records of the experts knowledge.

2.5. Different Type of Problem

Satellite anomaly resolution has significant differences from the anomaly resolution of most other systems such as aircraft. By design, satellites are highly reliable and have no physical access. Whereas most anomaly resolution for an aircraft involves isolating a common failure, most anomalies in a spacecraft are unique and unanticipated. Expert systems that utilize a knowledge base built up from experience and heuristics are inadequate for most satellite problems because the typical problem is new.

2.5.1. High Reliability

Many studies have been performed to attempt to anticipate problems before they occur, thereby decreasing the dependence on the skilled experts. Unfortunately, in spite of these efforts to anticipate problems and provide procedures and equipment to resolve them, as many as 90% of all major anomalies experienced by current satellite systems have been unanticipated and required experts for resolution (Figure 1). This is probably due to the great emphasis on system reliability. That is, if a failure mode is identified, the system is designed to make the failure extremely unlikely. Thus, the failures that occur tend to have not been identified before

hand.

2.5.2. Limited Access

Even though today's satellites are very complex, the limited number of actions that can be taken for any one situation creates a bounded problem. Although permutations cause this number to be large, it is much smaller than the number of possible actions that could be taken with a system such as an aircraft because the aircraft is physically accessible. Thus, space systems allow the use of techniques that would be inappropriate in other domains.

3. Potential Solutions

All of the problems discussed can be alleviated if satellites can be made more autonomous and thereby reduce the dependence on available human experts. The need for more autonomous satellites has been acknowledged by the Satellite Control Architecture Study sponsored jointly by Air Force Space Command and Air Force Systems Command Space Division. Actually, satellite autonomy is a function of system design, hardware reliability, redundancy, the environment, and a capability to analyze and act on changes in situations.

Fortunately, new software techniques associated with Artificial Intelligence (AI) research offer the unique hope of addressing the problem. The goal is to have a system that will act as an expert would in the same situation. To be truly successful, such a system must not be limited to predefined responses to predefined situations. It must, instead, be capable of reasoning about a new situation as a real expert would.

3.1. Redundant Components

Redundant components are an essential part of any satellite autonomy concept. Without redundancy, few, if any options exist for dealing with satellite anomalies. The traditional approach for employing redundancy is to provide automatic anomaly detection and automatic switching from the anomalous component to an equivalent backup. For some anomalies, automatic switching without understanding the nature of the problem may be necessary because of the time-critical response required. One example might be switching to a backup voltage regulator because of a loss of power. Unfortunately, automatic switching is based upon assumptions which may not be valid at the time the anomaly occurs. As a result, switching to a backup component can further complicate the situation. Thus, it's important to minimize switching to backup components when the anomaly is not fully understood. Because of limited ground support, the tendency for today's satellites is to employ automatic switching whenever possible, as long as further complications are not expected. In contrast to this method of using redundant components, an on-board self-reasoning software system could approach the problem at various levels to understand the cause of the problem much

ANOMALY HISTORY

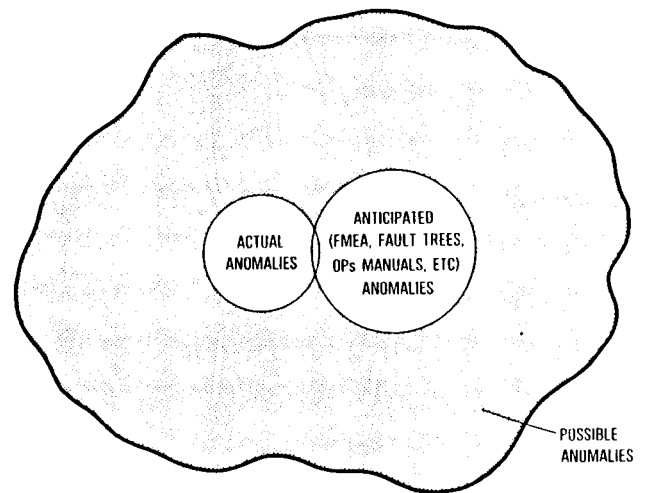


Figure 1.

as a human expert would.

3.2. Reliable Components

Today's satellites are generally composed of highly reliable components, and further advances in reliability will certainly enhance autonomy. If fact, if a component is considered reliable enough, the redundant component may be left out of the design. Nevertheless, some components are too critical to avoid redundancy no matter how reliable they are.

Another aspect of reliability is the impact on the expert supporting the satellite. A primary source of knowledge for the expert is the technical documentation for the satellite. However, it's also essential that the expert have operational experience in dealing with satellite anomalies in order to remain competent. The irony is that, while both a reliable satellite and a competent expert are desirable, the expert's competency depends, in part, on satellite failures!

3.3. Expert Systems

Many expert systems are in use today whose problem-solving performance matches or exceeds that of a human expert within a limited domain. By incorporating heuristics, or "rules of thumb," derived from a human expert(s), expert systems are able to mimic the behaviour of the human expert. These heuristics are the aggregate of experiences from which the expert has learned (sometimes the hard way). Often the expert is unable to precisely delineate a heuristic, or, even worse, will falsely delineate a plausible heuristic. Even without these problems, heuristics axiomatically reference predefined problems. Thus, an expert system by itself cannot provide autonomy for a satellite suffering unexpected problems. At best, an expert system can provide a reasonable initial attempt to isolate a problem.

The human expert, on the other hand, can fall back upon more general knowledge when simple heuristics don't solve the problem.

3.4. Neural Networks

In the area of health/status, Neural Networks (Neural Nets) have been used to diagnose satellite anomalies. The diagnosis can be almost instantaneous if the Neural Net is implemented with parallel processing. However, Neural Nets require many training examples of predefined problems and, like Expert Systems, cannot diagnose a problem that was never defined. Also, the cost of generating numerous real or precisely-simulated anomalies for the Neural Net to train on may be prohibitive. Even if the Neural Net correctly diagnoses the anomaly, it doesn't offer the potential for prescribing a solution.

In the mission area, Neural Nets may prove valuable in certain types of mission data interpretation (e.g., satellite photos). However, they don't appear to have the potential for autonomously allocating mission sensor resources to meet dynamic requirements.

3.5. Model-based Reasoning

It was stated above that AI promises significant promise to solving the described problems. Even though, the term "Artificial Intelligence" means different things to different people, several AI concepts appear to be well suited to the problem of satellite autonomy. The first of these is the idea of model based reasoning. Model based reasoning uses causal models of the system and its environment to reason about situations. The models are built using the object-oriented programming techniques. This allows concise models to be built for different physical aspects of the system (e.g. electrical models, structural models, thermal models, etc.). Each of these models would include a deep basic understanding of the specific satellite design as well as basic physical principles. Although the final models would be specific for an individual satellite, their architecture and much of their basic knowledge is generic. Finally, they will be built with tools making it easy to adapt them from one application to another. Model based reasoning uses special techniques to determine the cause of conflicts between model predictions and actual observed events. In addition, the models are used to construct solutions to problems and to try the solutions (through simulation) before actual commands are given to the satellite. Overall the model based concept provides (for the first time) the capability to deal with unanticipated events. The system would be able to reason about variances between the observed world and the world predicted by its models. The models themselves could then be modified to more closely match the observed world. They would also serve to evaluate a goal-oriented search toward resolving any problems. The resolution would not be limited to only predefined actions but could also be new, novel actions. It would also be much more flexible. Thus, when a power system was degraded,

the models would automatically adjust themselves to the new situation and would then serve to revise the constraints in power utilization procedures/schedulers. Viewed another way, the model based system tries to capture how the system should work as opposed to the traditional method of attempting to capture all possibilities of all problems. It's important to note that model-based reasoning is not limited to naturally occurring events. Hostile events that degrade or impact the system performance can be handled the same way. The system need not know all possible hostile actions, but merely that the world is not as it should be.

3.6. Natural Language

Another AI technique that is applicable to autonomous satellites is natural language. In its purest form, Natural Language programming would allow a user to converse with a machine in the same manner in which people communicate. Actually the current state-of-art for natural language is not yet that advanced. Although current language parsers are extremely useful in some applications, they are still only valid for relatively small domains. Natural language research is, however useful for applications other than natural language processing. This research has developed useful knowledge representation schemes, search techniques, and problem solutions. Some of its techniques for treating sequential events are of particular interest to satellite operations. Scripts is a programming technique that was developed for natural language understanding. It was found to be impossible to understand language without understanding the context of the situation. Words, sentences, and whole thoughts require an understanding of the situation to eliminate ambiguities. Scripts can be written for common situations to establish a loose relationship between events. These scripts can be referenced to understand the dialog. A similar concept can be used to understand the telemetry data of a satellite. For example, a script could be written to cover the events that usually occur when going into an eclipse or for some possible hostile events. These scripts then serve as possible references to resolve ambiguities in the telemetry data. A key feature of the scripts is that they serve as a reference framework and not as a mandatory sequence of events. This gives them great flexibility.

4. RADC Satellite Autonomy (SA) Program

4.1. Background

RADC has undertaken a major effort in the area of Artificial Intelligence, initially concentrating on tactical and intelligence applications. During 1985-86 RADC worked with the Air Force Satellite Control Facility and Space Division to fund some studies and research into the application of AI into satellite systems. In addition, Aerospace Corporation has done some research in the same area. Finally, many aerospace contractors have also been conducting independent research and development in this same

field. All of these studies concluded that artificial intelligence techniques combined with advances in computer speed, memory density, and architecture promise significant progress toward solving satellite autonomy problems.

Studies by Space Division and Space Command also defined roadmaps for future satellite control systems and thus help show how to incorporate the technology advances. These studies advocate a phased approach toward achieving autonomy beginning first with systems that interact with humans.

4.2. Description

Rome Air Development Center (RADC) has worked with the Strategic Defense Initiative Organization (SDIO), Air Force Satellite Test Center (AFSTC), Air Force Space Technology Center (AFSTC), Space Division (SD), Air Force Astronautics Laboratory (AFAL), and Air Force Space Command to create a program that uses AI techniques to achieve satellite autonomy. The program is a multi-phased effort that first shows the feasibility of the concept and then builds a prototype. This will be done using limited ground systems and then full ground prototypes for an existing satellite. The long term goal of this program is an on-board autonomous design. This goal includes both health/status and mission functions. The output of the program will be a system design that can be given to a System Program Office for incorporation into a new satellite design. This program is not designed to extend the AI technology, but it will make use of the most current technology methods and ideas. This is not simply a program to develop better built-in-test or data compression/data analysis, but is a program to develop a generic system capable of reasoning about itself or its environments.

4.3. Goals

4.3.1. Handle Unanticipated Situations

The ability to handle the unanticipated and the ability to generate novel solutions are key issues. This can best be explained by example. A typical function currently performed by people might be the scheduling of power utilization. Certainly, an algorithm could be made to perform this function, but as time progresses the satellite capabilities might change (systems fail, etc.) or the operational requirements may change. Thus, the real key to autonomy in this case is the ability to modify the scheduling procedure as events change. That is, a system should be able to develop and implement new scheduling constraints as the situation dictates. This is particularly true for situations where multiple users are tasking the same system. Therefore, a good scheduling system should be a flexible, reasoning type of system.

4.3.2. Generic Solution

The RADC SA program will be developed and demonstrated initially for three different types

of satellites. However, the emphasis is to prove the viability of particular advanced software techniques that can be applied to any satellite. These techniques can then be incorporated into the initial design of a satellite so that, once deployed, elaborate ground support is not required.

4.3.3. Cost Reduction

Autonomous satellites that can, among other things, recover to a maximum extent from anomalies, respond to contention for resources from users, and adjust nominal operations based on degrading components will provide a tremendous cost savings in terms of manpower and facilities. Note that the total numbers of people and facilities to support satellites will likely increase in the foreseeable future, even with highly autonomous satellites. This is because the total number of operational satellites will dramatically increase. However, the cost reduction goal of the RADC SA program is to reduce the manpower/satellite and facilities/satellite ratios (Figure 2). This will make future operations both possible and affordable.

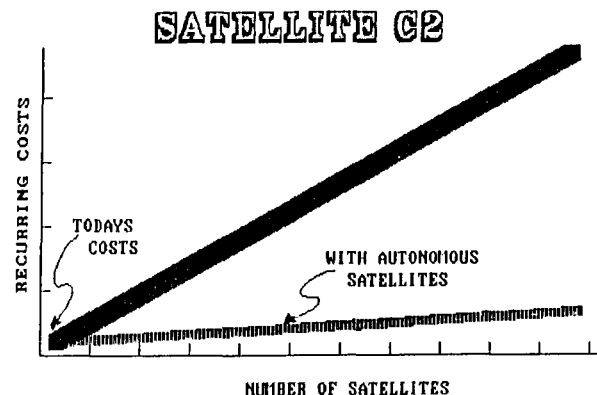


Figure 2.

4.4. Approach

Even though health/status and mission operational concepts can be considered separately, some integration and dependency exists. For example, mission sensors have telemetry which must be analyzed in the same manner as support function telemetry. Thus, both health/status and mission must be considered if greater autonomy is to be achieved.

One can reasonably expect components comprising the health/status and mission areas to improve in both reliability and redundancy. However, the RADC SA program will pursue greater autonomy through a synergistic application of expert systems, model-based reasoning, and natural language concepts as previously discussed.

4.5. Unresolved Issues

As described above, advanced software

techniques appear to offer promise toward solving the difficult problem of autonomy. However, there are some issues that must be explored before these techniques can be dictated for use on future satellites. Some of these issues are described below.

4.5.1. Scaling

The first of these issues is scaling. Although small self reasoning model-based systems have been built it is not yet proven that the results on these small systems can be scaled to larger systems such as an entire satellite. Fortunately, as stated above, the satellite problem is bounded. In addition, hierarchical structures and parallel processors may be used if scaling becomes a major problem.

4.5.2. Satellite Configuration

Another issue concerns the configuration of the satellite. A model-based system must always track the current configuration of the satellite including the status of expendables. Unfortunately, the exact status of all systems on the satellite may not be known (especially if anomalies have occurred). Thus, the model-based system may have to infer the status of systems. Since multiple inferences are often possible, the model-based system must be able to track multiple configurations simultaneously. As further information becomes available, the models must be revised and incorrect representations must be eliminated.

4.5.3. Data Archiving

Data archiving is yet another issue. Proper interpretation of events requires the system to archive data for later use. This is especially necessary for detecting gradual changes in system performance. Archiving can become extremely expensive in memory and obviously not all data can be archived. The problem lies in deciding how much to archive, how to compress it, and how to purge it.

4.5.4. Environment Modeling

Modeling the external environment will be necessary for a robust model-based system design. However, the external environment is not nearly a straightforward model as the satellite itself. Ideally, the external environment would be represented in enough detail to resolve environmentally induced situations, but not in so much detail as to unnecessarily complicate the system. Early studies have shown the advantage of supplementing a model-based system with an expert system (to heuristically handle anticipated situations) and this will probably be necessary for models of the environment.

4.6. Status

Currently, the Satellite Autonomy program is in its first phase. The three prime contractors are Boeing, Ford Aerospace, and TRW. These contractors are developing a system to prove the

feasibility against real satellites. In this first phase, a software system capable of doing all health/status for three major satellite subsystems will be developed. In addition, a system will be developed to perform one major mission function in this phase. These systems will be tested against existing robust satellite simulators and evaluated by current operational users. The exact designs for satellite autonomy are currently being developed, but several things have been shown:

4.6.1. Goal Attainment

The perfect software that can totally mimic the human expert is not yet in sight. However, it does appear possible to achieve most aspects of each of the goals described above. In addition, the software developed will exceed the human expert in many situations. This occurs because the systems are more thorough and can reason faster than people. Preliminary systems are addressing the key problem areas. They have found novel solutions to problems and have been able to reason about unforeseen situations.

4.6.2. AI versus Conventional Processing

An autonomous system will ultimately use both conventional and AI techniques. The problem is large and complex, but it still appears to be generally solvable.

5. Summary

The problems with operation and control of satellites are real, here today, and getting worse. Advanced techniques offer unique promise in this area. Although these techniques have limitations and concerns, they will go a long way toward solving these problems. The RADC Satellite Autonomy Program is a challenging program to use advanced techniques to develop a self reasoning system that can reduce the experts/system ratio currently necessary to operate and control satellites.

ORIGINAL PAGE IS
OF POOR QUALITY

GARBAGE COLLECTION CAN BE MADE REAL-TIME AND VERIFIABLE

James H. Hino
Integrated Inference Machines, Inc.
1468 East Katella Avenue
Anaheim, CA 92805

Charles L. Ross
Integrated Inference Machines, Inc.
1468 East Katella Avenue
Anaheim, CA 92805

ABSTRACT

An efficient means of memory reclamation (also known as Garbage Collection) is essential for Machine Intelligence applications where dynamic storage allocation is desired or required. Solutions for real-time systems must introduce very small processing overhead and must also provide for the verification of the software in order to both meet the application time budgets and to verify the correctness of the software. This paper proposes Garbage Collection techniques for symbolic processing systems which may simultaneously meet both real-time requirements and verification requirements.

The proposed memory reclamation technique takes advantage of the strong points of both the earlier Mark and Sweep technique and the more recent Copy Collection approaches. At least one practical implementation of these new GC techniques has already been developed and tested on a very-high performance symbolic computing system.

Complete GC processing of all generated garbage has been demonstrated to require as little as a few milliseconds to perform. This speed enables the effective operation of the GC function as either a background task or as an actual part of the application task, itself.

INTRODUCTION

Scientists and engineers may argue over the true nature of intelligence; whether it be human or artificial.

However, there is little argument that the capture and recording of useful knowledge does require more memory and more storage space than does the relatively straight-forward representation of numerical or alpha-numeric data entries.

The practical application of new Machine Intelligence technology to today's and tomorrow's Aerospace and Defense problems has become an important strategic issue to all of us. Application software programs are, therefore, becoming larger and more complex.

We must be concerned with the high-priority problems of (1) developing effective software which can perform its tasks quickly enough to meet demanding mission requirements; of (2) developing these programs in a timely and affordable manner; and of (3) verifying the correctness and the predictability of the final operational programming.

Simultaneous solutions to all of these needs is extremely challenging. The first priority need must be meeting the anticipated mission requirements. Those requirements include the capability of combined application software / hardware processor systems to produce essential information in time to make critical decisions or to control dynamic processes.

Real-time, knowledge-based systems programs, in particular, must accept a wide variety of types of data, including both numerical information and non-numerical information.

Non-numerical or symbolic data representations can easily include data items and associated data values which can vary enormously in terms of memory storage needs. A considerable waste of available real system memory capacity can occur unless dynamic memory allocation of variable size memory blocks is supported. Several languages including C, LISP, and ADA allow for dynamic allocation and de-allocation of memory.

For C, this task is left up to the programmer to handle as a part of the creation of the application software. In LISP, the task has been assigned to the designers of the LISP environment (includes the operating system) for a particular processor. This choice has off-loaded this demanding task from each individual programmer. Thereby reducing the possibility of unanticipated program flaws from this potential error source.

In the ADA language, the assumption is made that either the application program or its operating system (or both) may perform the memory reclamation (Garbage Collection) function, since the ADA Language Reference Manual does not specify that an ADA implementation must handle it.

Early experiments to allow programmers to allocate and de-allocate storage in LISP was disastrous. It proved to be extremely difficult for the programmer to know when all necessary data items are no longer referenced by any system process, program, or other data item. Some of the more intractable problems found in some C programs may be a by-product of the reliance on the application programmer to program this function, without leaving an unsuspected trap under certain program conditions. When large, complex programs are written by many individual programmers, the risk may substantially increase.

Some of the extra power and flexibility of the LISP language adds to the creation of considerable temporary results in main memory. Much of which is quickly no longer referenced and is, therefore, no longer required. This increases the importance of solving the GC problem. If memory is not reclaimed, free memory locations will soon become unavailable and program execution will stop.

This paper describes an approach to the design of a real-time GC mechanism. The proposed approach was demonstrated in the demanding LISP environment. It should be effective for ADA, as well. The performance tests were run using an implementation of the design for a uniprocessor architecture. The results should be appropriate for single processor systems or a system consisting of several individual processors, each of which are running separate application programs which co-operate together to meet a collective series of concurrent mission information processing needs.

The described approach may or may not be directly transferrable to the design of a multi-processor system, or be optimum for such a configuration. Additional on-going research will assess such feasibility and effectiveness.

STORAGE MANAGEMENT

Both data and program statements in LISP are represented in terms of symbolic expressions (S-expressions). S-expressions often appear as lists of items enclosed in parentheses. An S-expression is either an "atom", a list of S-expressions, or a "dotted-pair" of S-expressions. An atom is either a "numeric atom" such as an integer or a floating point number, or a "literal atom" which is a string of characters beginning with a alphabetic letter and containing other letters, digits, or a few other characters.

Atoms may be put together to form more complicated S-expressions using either a dotted-pair construction or a list construction. List construction is far more common in actual usage of LISP. S-expressions can be further combined with other S-expressions to build larger ones. Table I shows examples of a few of the various types of symbolic expressions, along with the definition of a symbolic expression and a list.

A fundamental assumption of LISP is that at any point in a computation process all memory cells (containing either programs or data) are reachable through a chain of pointers from a fixed set of known cells or base registers. Garbage Collection approaches must deal with the extensive series of relationships of data and programs which can exist at any time.

**ORIGINAL PAGE IS
OF POOR QUALITY**

**INFORMATION
UNAVAILABLE**

MEMORY RECLAMATION APPROACHES

The three basic forms of memory reclamation are "Mark and Sweep", "Copying Collection", and "Dynamic Pools". The following sections will briefly discuss the advantages and disadvantages of each approach for implementing real-time systems. The concept of a "workspace" is used in these discussions. A workspace is the collection of programs and data for any application as well as the entire system code. At any time the workspace may contain unreferenceable objects which is called "garbage".

MARK AND SWEEP

Mark and Sweep is also known as Stop and Collect. This technique requires that the processor perform successive passes through all of referenced memory. A specific data structure might be referenced several times. In the first pass, all accessible objects are marked. Then all marked objects are forwarded. The forwarding phase updates all pointers to their new locations. Finally, all marked objects are moved to their final destinations. Since objects are copied over each other, the application task may not run while the garbage collection is taking place. The collection process is activated when there is insufficient free memory to allocate an object or when requested by the application (a forced or commanded GC procedure).

This process must be performed over all modifiable objects. The process is verifiable in time and correctness if a forced garbage collection is commanded at a predictable place in the application program. A forced garbage collection is often desirable with this type of collection since the time required to collect garbage increases with the amount of garbage in the workspace.

A drawback to the Mark and Sweep technique is that the time required for most machines can take many seconds or even many minutes. Hardware support for garbage collection functions, on even fast machines, has typically still fallen short of the requirements for real-time applications.

COPYING COLLECTION

Copying Collection is a popular form of memory reclamation used by several LISP machines. A copying collector splits memory into two parts, known as hemispaces. Accessable objects are copied from one hemisphere into the other, leaving a forwarding pointer behind in its place. When all accessable objects are copied, the direction of copying reverses. The process of changing the direction of copying is called a "hemispace swap". This method of garbage collection can better approach real-time since it only copies a small amount of memory at any given time wherein the application program is stopped.

A key problem associated with copying collection is that it introduces additional uncertainty into the application processing time. Performance can be unpredictable since the actual time required is dependent upon when a hemispace swap occurs.

Another parameter that affects the variability of processing time of an application is the amount of information that is being copied between hemispaces. This quantity is a function of how much memory is being utilized versus how much free memory exists, and is not constant over time.

The final aspect of copying collection that affects the variability of processing time is that when an object is moved, all references to that object must traverse an indirect pointer to reach the desired object.

It is generally thought that a Copying Collection approach requires less overhead than a Mark and Sweep technique since the Mark and Sweep process passes through memory three times. This is not necessarily true. It depends upon the implementation, and especially the effective use of tag bits available in a tagged architecture.

One final note of significance is the amount of memory required to implement a copying collection approach. Since the available memory must be divided into two hemispaces (a "FROM" Space and a "TO" Space), it can take up to twice the amount of heap memory as other GC approaches.

DYNAMIC POOLS

Languages where commands to deallocate discarded memory is required, can use a scheme where there exist dynamic pools of allocated and available memory. This scheme works well if allocations are of a constant size. If allocations are of varying sizes, memory fragmentation exists. Fragmentation will cause compaction to be required. The time for compaction is a function of how much memory is required and how memory is fragmented. The process of searching for free memory occurs at each allocation. This makes verification of time budgets difficult, if not impossible.

A NEW HYBRID APPROACH

Integrated Inference Machines, Inc. has designed and implemented a memory reclamation technique that takes advantage of the strong points of the Mark and Sweep and Copying Collection approaches. It has been called the SCORE GC. SCORE stands for Stop-and-Collect, Optimizing, Real-time, Ephemeral garbage collector.

The garbage collector is a callable microcode routine that is invoked by a special opcode. The collector utilizes tag bits associated with a hardware-supported tagged architecture machine. Two bits of the 8-bit tag associated with each word in memory is used to support the GC function.

The SCORE garbage collector separates memory into "Static Space" and Heap Space. All objects in Static Space do not move and the memory they occupy does not need to be reclaimed. All new objects are allocated from "Free Space" into "Heap Space". Objects in Static Space need not be read-only, but when they are modified to be a pointer to an object in Heap Space, the address of this location must be saved for the garbage collection process. Figure 1 shows how memory is partitioned.

INFORMATION UNAVAILABLE

To begin the Mark Phase of a garbage collection procedure, the list of objects in Static Space is added to the "Mark Seed" (the mark seed is typically the execution stack as well as objects required to handle asynchronous events such as errors or interrupts). This is required to ensure that objects in Heap Space that are only referenced from Static Space are properly marked. All objects in Heap Space are then collected, using a modified Stop and Collect algorithm. The mark phase is also modified to stop if an attempt to mark an object in Static Space is made.

The Forwarding Phase operates, normally, on Heap Space only. This algorithm provides compaction which can make objects move. Therefore, when Heap Space is forwarded, the locations in Static Space that reference objects in Heap Space are forwarded to reflect the new location of the object in Heap Space. Finally, the Compaction Phase operates normally on Heap Space.

The setting of the boundary between Static and Heap Space, and the timing of the garbage collection process is a function of the application. The optimal partitioning places unmodified objects and objects which do not contain pointers into Static Space. All other objects are placed into Heap Space.

For real-time applications, the timing of the garbage collection is forced by the application program on a regular basis. This collection is placed at the end of one iteration of the application. This is typically where required references to temporary objects is at a minimum.

In order to meet the requirements of real-time applications, speed as well as verifiability of performance is required. SCORE memory reclamation does not occur in the background. Repeated timings of an application, as well as the time to reclaim garbage is repeatable, down to the number of machine cycles. Every run of an application with a given environment of data inputs is identical to the previous and to the next. Memory locations are not altered in ways that cannot be reproduced.

The SCORE GC also can be operated in an ephemeral mode. In this mode, garbage collection is performed when a small amount of memory is used. As objects survive collection, they move into Static Space. Intermediate spaces can be added with different rates of collection to provide additional ephemeral quality.

PERFORMANCE RESULTS

A single SCORE GC cycle has a minimum runtime of under 2 milliseconds. The SCORE collector requires a very small percentage of processing time to perform its functions. An average GC overhead of approximately five percent of the application runtime is predicted. The average number is useful since some tasks may create little or no garbage, while others will generate a great deal.

Of equal importance, the resulting GC overhead for a given application is measureable and is repeatable. Tables II, III, and IV show the results of GC tests performed by the author using a very high performance symbolic computer, the SM45000 (developed and manufactured by IIM).

Table II identifies a series of benchmark programs from the Gabriel Benchmark suite used for the evaluation. The suite contains benchmarks known to produce little or no garbage as well as ones which produce a significant amount of garbage. The benchmarks were timed for conditions of No Garbage Collection overhead at all (Table II), operation of the SCORE GC in real-time mode (Table III), and operation of the SCORE GC in the ephemeral mode (Table IV).

**INFORMATION
UNAVAILABLE**

**INFORMATION
UNAVAILABLE**

CONCLUSION

The power and flexibility of dynamic memory allocation and de-allocation can be made a part of real-time systems. Memory reclamation (garbage collection) technology has advanced to the point where fast, predictable memory management processing can accommodate these requirements. Verifiability of the resulting dynamic memory application software does not have to be sacrificed to an essentially background processing task which can, in turn, alter the dynamic memory states and defy repeatability.

The full performance of the technique makes significant use of two of the tag bits within the 8-bit tag field associated with each 32-bit word in memory.

The absolute GC processing times can be reduced still further by speeding up the symbolic processor, itself.

**ORIGINAL PAGE IS
OF POOR QUALITY**

ORIGINAL PAGE IS
OF POOR QUALITY

AN ARCHITECTURE FOR INTEGRATING DISTRIBUTED AND COOPERATING KNOWLEDGE-BASED AIR FORCE DECISION AIDS

Richard O. Nugent
Nugent@MITRE.Arpa

Richard W. Tucker
RWTucker@MITRE.Arpa

The MITRE Corporation
Washington C³I Artificial Intelligence Technical Center
7525 Colshire Drive
McLean, Virginia 22102

ABSTRACT

MITRE has been developing a Knowledge-Based Battle Management Testbed for evaluating the viability of integrating independently-developed knowledge-based decision aids in the Air Force tactical domain.

The primary goal for the testbed architecture is to permit a new system to be added to a testbed with little change to the system's software. Each system that connects to the testbed network declares that it can provide a number of services to other systems. When a system wants to use another system's service, it does not address the server system by name, but instead transmits a request to the testbed network asking for a particular service to be performed.

A key component of the testbed architecture is a common database which uses a relational database management system. The RDBMS provides a database update notification service to requesting systems. Normally, each system is expected to monitor data relations of interest to it. Alternatively, a system may broadcast an announcement message to inform other systems that an event of potential interest has occurred.

Current research is aimed at dealing with issues resulting from integration efforts, such as dealing with potential mismatches of each system's assumptions about the common database, decentralizing network control, and coordinating multiple agents.

INTRODUCTION

Integrating heterogeneous software systems is a burgeoning problem, particularly for the military. Many independently-developed systems produced for the military are stand-alone decision aids. This paper describes an architecture which supports the integration of such command and control (C²) systems and discusses the required characteristics which enable these systems to cooperate and share information with each other.

MITRE's Knowledge-Based Battle Management Testbed has been the vehicle for performing experiments in

integrating knowledge-based systems for the Rome Air Development Center (RADC) [5]. The testbed employs a core set of functions which provide control mechanisms and open connectivity support, called the knowledge-based battle management (KB-BATMAN) shell. The type of systems for which the testbed is intended are coarse-grained, loosely-coupled systems. A coarse-grained system has a large amount of functionality, and a loosely-coupled system has a high level of independence from other systems; such a system does not require a great deal of communication with external agents, and can act autonomously most of the time. A primary goal of the testbed architecture has been to permit systems to be able to "plug in" dynamically and even be replaceable by systems offering similar functionality.

Three realistic Air Force tactical C² systems operate in the current testbed: a mission planner, a simulator, and an intelligence analysis system. The goal of the testbed project has been to link these three coarse-grained systems using the KB-BATMAN Shell and to determine what problems must be addressed to assure effective cooperation among them. The principal way in which these three systems are linked is by relaying outputs from one system, such as the intelligence analyst, to become inputs to another system, such as the mission planner. This concept of integration seems simple; however, a variety of issues are involved, some of which have been addressed during the testbed project.

The principal issues that have been addressed include how to control cooperation, how to permit commonly-used information to be used by several systems, and how to deal with different views or representations of information.

BACKGROUND AND PROBLEMS

Related Research

It is difficult to evaluate the effectiveness of a decision aid in isolation from other systems with which it may interact. Graham describes a model for representing the interaction of systems with their environment, where the environment is the essential "glue" through which the systems interact [4].

Graham views the environment as one more system to be modeled in a distributed simulation of C² systems.

Previous MITRE work on the AirLand Loosely Integrated Expert Systems (ALLIES) project [1] involved integrating an Army planning system, an intelligence analysis system, and a simulation system into a single cooperating environment. Since these systems were integrated after each was developed to operate stand-alone, the methodology for integration was ad hoc and communications required several different protocols.

A better environment for developing cooperating, distributed systems is essential to encourage modularity of system design and to provide well-defined interfaces among systems. Teknowledge, Inc. has been developing ABE for RADC and the Defense Advanced Research Projects Agency (DARPA) to meet these goals [3]. ABE was not used in the testbed project because it was still under development when evaluated.

Integration of heterogeneous decision aids requires addressing issues involving the fields of distributed computing, databases, networking, and knowledge-based systems, among others.

Distributed Control

In any distributed environment, control of intersystem activities may be centralized or decentralized, or a hybrid. Centralized control is easiest to implement, but also provides a single point of failure, which would not be desirable in an operational system in most cases. Decentralized control requires fairly complex algorithms for coordination of systems. We use centralized control in our testbed, in part to keep the architecture simple, and also to support monitoring of testbed communications.

Common Functionality

While heterogeneous software components should be loosely-coupled to prevent each system from becoming highly dependent on other systems, there still is a need for sharing information that is not specific to a single system. Two common systems have been identified to fulfill this requirement: a Common Database manager and a Common Knowledge Base manager. These two components are considered to be integral parts of the KB-BATMAN Shell, although like other systems in the testbed, they are modules which can be replaced without affecting other systems.

A relational database management system (RDBMS) is used for the Common Database because the functionality of RDBMSs are fairly standard and implementations are available for a wide variety of computers. Most properly-designed decision aids should have an easily identifiable set of database access functions which may be replaced with RDBMS functions accessing a Common Database.

When a new system is being integrated into the testbed, it is important to determine how it uses a

database. One problem is to decide which data elements are of interest to other systems in the testbed, and which data elements are for internal use only. A second-order problem is to ascertain how to translate data representations into a form that is most appropriate for access by multiple systems, since different systems may view the same collection of data in different ways. Each system's view of the organization of data must be transformed to the Common Database's actual view. To solve this problem, the Common Database system must be able to provide an intelligent database viewing mechanism, in which a database request from a system may be translated to a combination of select, join, and project operations in order to provide the requested view. The alternative to providing intelligent interfaces is to modify the internal structure of a system, which is likely to be an undesirable option for large-scale, coarse-grained systems.

There are further issues resulting from multiple access to commonly-used information. One system may use a different method of representing some entity; for example in the testbed, one system uses latitude and longitude to identify a ground location whereas another system uses the universal transverse Mercator coordinate system. Also, one system may be interested in greater precision or detail for some data than needed by another system. Interpretation of uncertainty qualifications to data is likely to be difficult or impossible to correlate between systems.

The Common Knowledge Base includes commonly-needed behaviors or functionality for the Air Force problem domain. It can be used to reduce the duplication of effort in component systems. It can also be used to enforce standard operating procedures as well as Air Force doctrine. Further work remains to be done on the Common Knowledge Base, particularly for its potential role as an overall director for a suite of C² systems.

Impact on Using Existing Systems

An early goal of the testbed project was to address the issue of using existing decision aid systems. It is impractical to suggest that any existing system can be easily adapted for integration into our testbed architecture. Systems which were not designed with integration in mind are especially likely to be difficult to adapt. It may be more cost-effective and reliable to reimplement a system to fit the architecture than to patch existing software.

THE KB-BATMAN SHELL ARCHITECTURE

Message Passing

Testbed components communicate with each other by sending messages. Three types of messages are used: a request, a reply, and a notification. A request corresponds to the concept of remote procedure call from distributed computing. A reply contains data in response to a request. A notification is an announcement which does not imply that a reply is expected.

ORIGINAL PAGE IS
OF POOR QUALITY

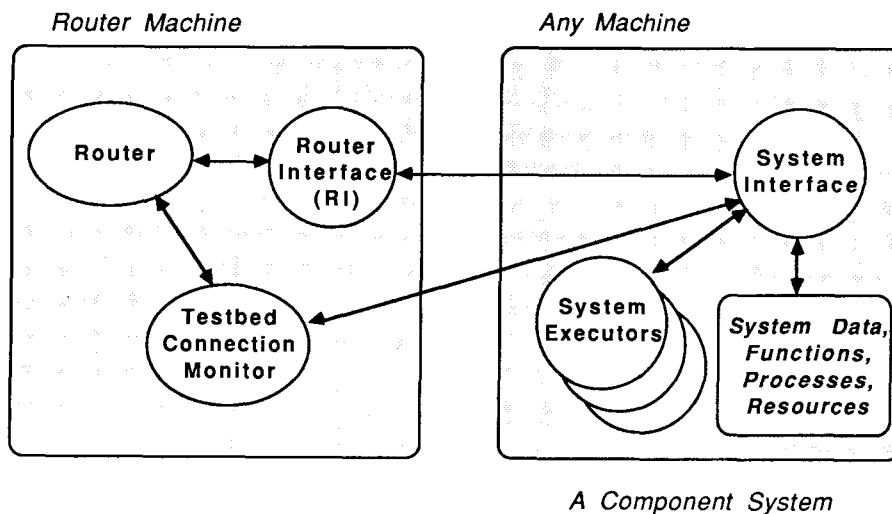


Figure 1

Processes

Multiple processes are involved in the KB-BATMAN shell concept. The following types of processes are used in the testbed:

- The Router
(one process)
- Testbed Connection Monitor
(one process)
- Router Interfaces (RIs)
(one per hosted system)
- System Interfaces (SIs)
(one per hosted system)
- System Executors
(one per active request execution)

Figure 1 shows the interrelationships among these processes

The Router supports centralized control of communications among hosted systems. All messages pass through the Router. Figure 2 depicts the star network of systems communicating through the Router.

A Router Interface and System Interface together form the communications interface between a system and the Router. The interface consists of two parts because the system may execute on a different computer than the Router. A system's RI process executes on the same machine as the Router; the SI process can operate on any computer, but typically is associated with a machine representing the hosted system. (A hosted system itself may operate on multiple computers). Processes on both computers need to poll for message arrival from either side, either from the system or the Router.

The Testbed Connection Monitor operates on the same computer as the Router. Its purpose is to handle requests from systems on other machines to connect themselves into the testbed. The Connection Monitor asks the Router process to create an RI for the system. The RI and SI then will open the necessary network connections to support message passing.

Through its SI, each system declares to the Router the services it can perform upon request. For example, the Common Database system advertises that it will service database access and database update notification requests. In addition, "declare services" is a built-in service handled by the Router.

When a system's SI receives a request for one of its services from another system via the Router, it executes that request by evaluating a function asynchronously in a separate System Executor process. In other words, the service request can be in execution while the SI continues to poll for further messages; in fact, multiple service requests can be in execution, each in a separate System Executor process.

The Router's role is to maintain a "yellow pages" of all declared services. It is possible that a service can be performed by more than one system. When a system requests an external service, its SI sends the request to its RI which relays it to the Router. The Router determines which system is most appropriate to perform the service by selecting one from the set of all declared servers. (In the present implementation, no criteria are applied for selecting from multiple servers. Criteria might include speed of response, accuracy of response, currency of data, etc.) The service requester does not address its request to a particular system; in fact, the requester does not know what system, if any, will perform a service. It is possible that a service is not supported, in which case the Router sends an error indication as a reply to the requester. The Router currently does not interpret the contents of messages.

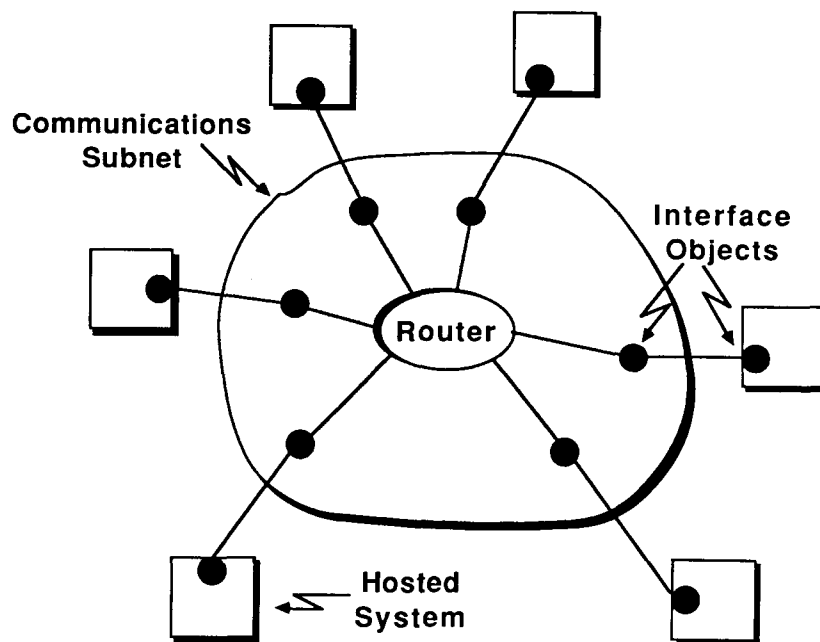


Figure 2

A System Executor process is created by an SI to perform a requested service. The System Executor can be considered to be performing the service in the context of the SI's system since it has access to any functions and data of that system. System Executors are implemented as reusable process resources on a LISP machine, so after one completes the servicing of a request it becomes eligible for reuse.

Consider a situation in which a service handling system does not respond to a service request in a timely fashion, either due to system failure or because it is busy doing other things. The requesting system would wait a long time or even forever unless it is designed to monitor for replies and eventually time out. In general, it is preferable for distributed systems to be data-driven, responding to changes introduced by external sources (such as other systems or an operator), rather than function-driven, in which the system asks another system to perform a function and then waits for a response. A data-driven system is easier to coordinate with other systems than a function-driven one because it is reactive rather than dependent. Nevertheless, function-driven or service-oriented systems are necessary for a variety of general-purpose common functions that multiple systems may need to use, such as accessing the Common Database and using resource managers.

Input Ports

When a message is transmitted from one testbed process to another, it is stored in a process' input port. Each process other than System Executors has one input port, and the process may read messages from the input port in any order it chooses. Presently, most processes read and process all messages in a first-in, first-out (FIFO) manner. However, a System Executor process scans the contents of the input port

of its currently associated SI for whichever replies it is expecting. A System Executor does not process any request messages, and the System Interface does not process any reply messages. Each process which monitors a network connection for input will copy network input messages into its input port for subsequent processing.

TAC-2 APPLICATION SYSTEMS

TAC-2 is the name of the version of the testbed which incorporates three realistic knowledge-based systems in the Air Force tactical domain: a mission planning system, an intelligence analysis system, and a simulation system. These three systems were developed by three different groups of people at different locations. One, the planning system, was under development prior to the initiation of TAC-2, whereas the other two were developed expressly for use with TAC-2.

The planning system used in TAC-2 is the Air Force Mission Planning system (AMPS), which is a successor to the KNOBS Replanning System (KRS) [2] which in turn was a successor to the Knowledge Based System (KNOBS). All of these systems have been developed by the MITRE-Bedford Artificial Intelligence Technical Center independently of the TAC-2 effort underway at the MITRE-Washington Artificial Intelligence Technical Center. The concepts embodied in KRS have also led to the current development of TEMPLAR for use as a operational system by the Air Force.

AMPS was developed as a stand-alone system, and none of the AMPS software was developed for use in TAC-2. However, one component of AMPS, its relational database management system, was adapted for use as part of TAC-2's Common Database system. A number of enhancements were made to the DBMS in

ORIGINAL PAGE IS OF POOR QUALITY

order to support simple methods for remote access and update notification. The Common Database system is used to manage data for all component systems in TAC-2.

The intelligence system in TAC-2, INTEL, was developed by the same staff at MITRE-Washington that developed TAC-2 and therefore was easiest to adapt to the conventions required for inclusion in the TAC-2 testbed.

The simulation system in TAC-2, SIMULATOR, was developed by staff at the Rome Air Development Center. SIMULATOR was designed to work with TAC-2, but included some differences with the other systems in assumptions about data. For example, SIMULATOR assumes that location data is given in universal transverse Mercator coordinates whereas AMPS assumes that location data are given as latitude/longitude pairs. The transformation between the two location representations is complex. How can systems cooperate if they have differences like this? An intelligent interface to the Common Database must transparently supply the correct data to each system.

These three domain systems cooperate by reading and writing data to the Common Database. AMPS plans offensive counter-air (OCA) missions automatically based on target and other data present in the Common Database, built-in planning constraints, and optional user inputs. The Simulator simulates flying these missions, assessing bomb damage to the targets and loss of aircraft due to surface-to-air missiles (SAMs). The INTEL system prioritizes targets for bombing missions.

All TAC-2 component systems and support software are written in LISP (both Common LISP and ZETALISP) and run on Symbolics LISP machines. INTEL and SIMULATOR operate in the latest version of the operating system software, whereas AMPS executes in an earlier version. The KB-BATMAN shell software and the Common Database system software operate in both versions, using the same source code. Communications between LISP machines employs generic networking software, and can use either TCP/IP or Chaosnet protocols for transmission of messages between a system and the KB-BATMAN Router. TAC-2 can operate on as many as five computers, with each of the Router, Common Database, AMPS, INTEL, and SIMULATOR on a separate computer; or as few as two, with AMPS on one computer and the others all on another computer.

SUMMARY AND CONCLUSIONS

The KB-BATMAN Shell architecture provides support for integrating coarse-grained knowledge-based decision aids. The most important aspect of the architecture is the emphasis on maintaining independence of systems. Independent systems can be considered more manageable and robust than systems that rely on other systems for control directives. Independence is encouraged through the use of intersystem messages which are *not* addressed to specific systems. Instead, messages are either

service-oriented, to be relayed by a Router to a system which supports the service, or broadcast into the environment for all systems to examine. A message may be nothing more than a piece of data in a shared Common Database, in which case system control is totally data-driven.

Another key aspect of the architecture is the use of intelligent interfaces to systems. Intelligent interfaces can be used to adapt data from the external environment (e.g., the Common Database) to be in a form suitable for internal system use. These interfaces may need to employ knowledge-based techniques for transforming data from a common representation to a specialized one used within the system. If necessary, an interface can interact with the environment and other systems in order to support its system's needs.

Further work is required to address issue areas such as decentralizing control away from a single Router and supporting component connectivities other than a star network. The Router needs to be extended to permit servers to provide qualifications for providing a service, and on the other end, to permit servers to be able to preview a request to determine whether it is interested in servicing it. The latter improvement would be necessary for a totally decentralized control mechanism such as contract nets, in which systems bid on service requests.

ACKNOWLEDGMENTS

This document was completed under Air Force Contract F19628-86-C-0001 in support of the Rome Air Development Center, Griffiss Air Force Base, New York.

REFERENCES

1. Benoit, John W. et al., *AirLand Loosely Integrated Expert Systems: The ALLIES Project*, MTR-86W00041, The MITRE Corporation, McLean VA, April 1986.
2. Dawson, Bruce C., Richard H. Brown, Candace E. Kalish, and Stuart Goldkind, *Knowledge-Based Replanning System*, RADC-TR-87-60, Rome Air Development Center, Griffiss Air Force Base NY, May 1987.
3. Erman, Lee D., Jay S. Lark, and Frederick Hayes-Roth, *Engineering Intelligent Systems: Progress Report on ABE*, TTR-ISE-86-102, Teknowledge, Inc., Palo Alto CA, May 1986.
4. Graham, Richard A., *An Environment for Distributed Simulation of Command and Control Networks*, Master of Science Thesis, Naval Postgraduate School, Monterey CA, March 1983.
5. Morawski, Paul E., Richard O. Nugent, and Richard W. Tucker, *TAC-1: A Knowledge-Based Air Force Tactical Battle Management Testbed*, MTR-87W00171, The MITRE Corporation, McLean VA, September 1987.

COOPERATIVE PROBLEM SOLVING IN PILOTS ASSOCIATE

Scott Fouse
Teknowledge Federal Systems, Incorporated

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

A Parallel Expert System for the Control of a Robotic Air Vehicle*

Donald J. Shakley
Avionics Laboratory
Air Force Wright Aeronautical Laboratories
Wright-Patterson AFB, OH 45433

Gary B. Lamont
Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

Abstract

Expert systems can be used to govern the intelligent control of vehicles, for example the Robotic Air Vehicle (RAV) which is currently a research project at the Air Force Avionics Laboratory. Due to the nature of the RAV system the associated expert system needs to perform in a demanding real-time environment. The use of a parallel processing capability to support the associated expert system's computational requirement is critical in this application. Thus, algorithms for parallel real-time expert systems must be designed, analyzed and synthesized. The design process incorporates a consideration of the rule-set/face-set size along with representation issues. These issues are looked at in reference to information movement and various inference mechanisms. Also, examined is the process involved with transporting the RAV expert system functions from the TI Explorer, where they are implemented in the Automated Reasoning Tool (ART), to the iPSC Hypercube, where the system is synthesized using Concurrent Common LISP (CCLISP). The transformation process for the ART to CCLISP conversion is described. The performance characteristics of the parallel implementation of these expert systems on the iPSC Hypercube are compared to the TI Explorer implementation.

Introduction

Artificial Intelligence (AI) is concerned with the designing of computer systems that exhibit intelligent characteristics of human behavior. These methods are used when other direct approaches start to deteriorate due to a lack of generality of solution. Examples of such behavior include language understanding, reasoning, and problem solving (Barr and Feigenbaum, 1981). These problems are studied in AI by using a computational model. Many computational models exist for AI problems. A computational model is a formalism used to describe a method of solution. These models present different ways to represent the problem domain. Examples of these models include production systems, semantic networks, frames, and logic (Fischler and Firschein, 1987).

A specialized area of AI called expert systems development has had considerable success with a multitude of applications. Many applications use production systems or rule-based system structures employing commercial expert system shells. These structures apply heuristics to solving the problem along with algorithmic meth-

ods. The success of these expert systems is directly related to the quantity and quality of the associated knowledge base (rules and facts).

Real-time applications exist that involve "hard" problems that currently defy generic algorithmic approaches. Thus, problem solving paradigms from Artificial Intelligence (AI) are being applied to these applications using expert system structures. Due to the computational complexity, however, these approaches have poor computer performance characteristics (Gupta, 1986). Parallel processing seems to offer a possibility to improve expert system computational performance for hard real-time problems.

Parallel processing is the use of more than one processing element to compute the solution to a problem. By using more processing elements, it is hoped that the time to solve the problem is reduced over the time to solve the problem on a single processor. There are several ways to achieve performance improvements in computer systems besides parallel architectures: faster hardware technology, improved serial architectures, better algorithms and code optimization. There are several reasons, however, for looking toward parallel architectures. First, parallel architectures can evolve as fast as hardware technologies become available. Second, many problems associated with AI are computationally "hard" (exponential time-order) or NP-complete. If a problem is NP-complete, this implies that time-order improvements in solution algorithms are unlikely due to many years of computational studies (Aho, Hopcroft, and Ullman, 1974). It should be noted that parallelism can not produce polynomial time solutions to exponential time problems (Norman, 1985). But, it is possible to improve the constant term of the time complexity. Also, the exponential time bound is often times worst case. In AI problems, the use of heuristics can reduce the time complexity of state space searches in specific applications. Third, some problems seem to lend themselves to parallel solutions because the problems decompose easily into independent, computationally equivalent pieces. Production systems, for example, seem to fall into this category because of the large number of rules that must be matched during each production cycle.

This paper discusses real-time processing with application to the Robotic Air Vehicle (RAV) expert systems. Consideration is given to parallel search algorithms and associated knowledge-based structures in the design and implementation of a parallel processing expert system. Experimental and theoretical results are presented.

Real-Time Processing

There are several important issues in the analysis and design of real-time computer applications. One of

* Research supported by the Air Force Wright Aeronautical Laboratories and the Strategic Defense Initiative.

the important characteristics is the critical nature of the system execution speed in reference to external events. This can be viewed in terms of the response time of the system to a particular input. For a real-time system, "the time needed to make a calculation has to be less than the time from when the need for the calculation is recognized until the time when the response is needed to take action" (Norman, 1985). This can vary with the system, but the time is generally relatively small. Relatively small is definitely less than a second and often in the milliseconds or less (Ward and Mellor, 1985).

Another critical characteristic is limited memory capacity. Real-time software typically needs to run in an environment where the size of the program can become a problem. A third consideration is the correctness and integrity of real-time software. The system needs to run correctly and without failure a high percentage of the time (Ward and Mellor, 1985). These represent the most critical issues dealing with real-time systems. In addition, real-time expert systems must also focus on efficient memory interfacing, integration with specific application software processes, efficient inferencing mechanisms, and external temporal commands and events.

The problem with a real-time system on a serial architecture is that the execution time and space requirements are relatively fixed for a given operation. A desirable feature of a real-time system would be a variable time and space performance based on the need. With parallel architectures this could be possible. If a problem needed a faster solution based on the time requirement, then more processors could be added to produce the appropriate speedup. This could only be done if the speedup were predictable.

The need for production systems within real-time systems is growing. With parallel processing of production systems, the execution speed is increasing. For real-time systems this speedup needs to be predictable, so that at any given moment more processors can be brought to bear on a problem to decrease the coefficient of the time complexity of the solution.

An example of a real-time application, which is a current research project at the Air Force Avionics Laboratory, is the Robotic Air Vehicle (RAV). It is an air vehicle with the capability of autonomous flight operation. This vehicle needs the capability for the "intelligent" control of an air vehicle, the capability to plan and replan missions, and the capability to access flight data on various geographical locations (airbases, airports, cities, etc). By "intelligent" it is meant that the system can react to conditions rather than fly on a rigid preprogrammed flight path. A diagram of the system can be seen in Figure 1. This system is an example of hierarchical control which may permit real-time performance due to its decompositional structure. This is also known as meta-level expert system organization. Thus, the course granularity of the multiple expert system framework provides yet another source of parallelism.

This study focuses on the "intelligent" flight control components of the system. This component was selected for the feasibility study due to its reliance on production systems and its maturity in relation to the entire research project (Shakley, 1987a). The control of the vehicle can be thought of as a search through a finite state-space over a time period of the vehicle's operation. The problem of intelligent control of a robot is a control-type NP-complete problem that is best suited to be solved by a production system in real-time. Therefore, this system makes an excellent tool for the study of parallel AI search techniques for real-time applications.

The RAV system is an example of an intelligent

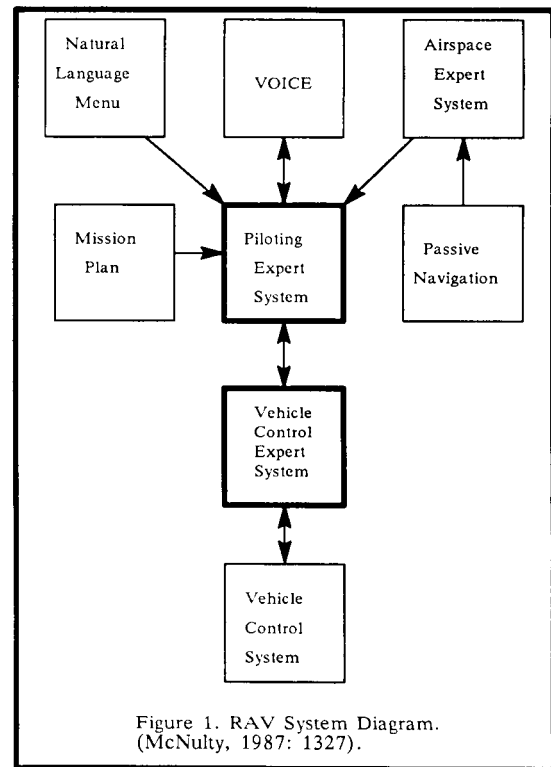


Figure 1. RAV System Diagram.
(McNulty, 1987: 1327).

real-time robotic control system implemented using an expert or production system (McNulty, 1987). The purpose of this investigation is to try to increase the performance of the expert system by reanalysis, redesign and reimplementation of the system on appropriate parallel architectures. The hypothesis of this study is that the performance of the RAV expert system can be improved in a predictable and linear manner.

This research is also intended to be a feasibility study of the various issues involved with implementing a parallel expert system. These include implementing an expert system written in Automated Reasoning Tool (ART) on a TI Explorer and on the iPSC Hypercube using Concurrent Common LISP (CCLISP). ART is a knowledge engineering language used in the development of expert systems. CCLISP is a dialect of Gold's Common LISP that has been enhanced to allow for message passing on the iPSC Hypercube. LISP was chosen since it was available on both the TI Explorer and iPSC Hypercube making the transportation of the code from one machine to the other easier. This study is most interested in examining the execution speed of real-time systems that use production system structures. Achieving execution "speedup" rests on the use of parallel algorithms. The results are not intended to specify final real-time execution times, but rather present an analysis of speedup possibility due to parallel processing of production systems.

The current knowledge base for the RAV has been obtained from TI through the Air Force Avionics Laboratory. This includes a basic demonstration routine. Portions of this demonstration are used to exercise the system. The control for the expert system is developed using the basic principles of production system control for an inference engine. The current RAV software uses the Automated Reasoning Tool (ART) as the inference engine (McNulty, 1987). ART can not be used with the parallel environment since it is not available for the iPSC Hypercube. The inference engine is implemented on the TI Explorer Lisp machines where it can be tested against the

knowledge base and the rule execution timing results can be compared to the ART inference engine. The parallel expert system is implemented on the Intel iPSC Hypercube with up to 32 processing elements (PEs) to explore larger degrees of parallelism.

Parallel Search

The advent of parallel computer architectures have addressed the possibility of faster execution of many computer applications. Parallel architectures have brought about new problems as well as the old in terms of software analysis and design. For an application to be implemented on parallel architecture, a way must be found to decompose the problem into component parts. Several important issues are concerned with this decomposition. First, the work must be *distributed* as evenly as possible for an equitable load balancing. Second, the *communication* between the pieces needs to be kept to a minimum. This is to reduce the communication overhead associated with the various processors communicating with each other. However, when this communication occurs, the processors need to be *synchronized* with respect to each other. This is to prevent problems with updating shared variables that can produce erroneous or unpredictable results. Proper synchronization also prevents the occurrence of deadlock between processors (Ishida and Stolfo, 1985).

Speedup is the most common performance measurement (metric) in parallel computing. This is the ratio of the run time of the concurrent software running on n nodes over the run time of the best serial solution. An application is said to be "perfectly parallel" or have a linear speedup if this ratio is n . Often the speedup approaches the linear speedup, but does not reach it due to the communications overhead between the processing elements (Gupta, 1986). Although rare, speedups have been observed greater than n . This is called *super linear speedup*. At first this seems to be absurd, but upon further study it does seem reasonable. Super linear speedup usually occurs when the application is so large on a serial system that certain overheads are incurred, but when placed on many processors none of the pieces is large enough to incur the same overhead. Thus super linear speedup is observed (Kornfeld, 1981). In addition, there is usually a point at which the addition of more processors does not improve the speedup (Gupta, 1986).

Communications overhead is a large concern in parallel computing. Communication takes several forms. The first is the time to set the job up on the parallel system or the time to distribute the work. The second involves the time needed to collect the results of the job. The third is the communication needed between the processors during the running of the job. An important measurement is the time a processor is communicating versus processing. This measurement along with the setup time and cleanup time gives a good indication of the overhead associated with the parallel process.

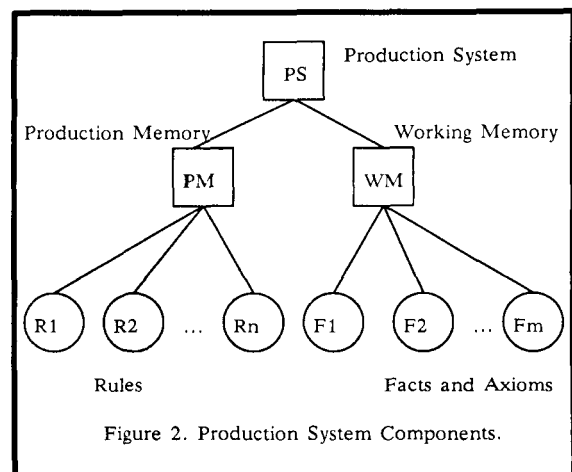
Load balancing is another important criteria for parallel computing. This is the percentage of the total processor power that is used during the job. A perfect load balance would be one in which all the processors are busy all the time. This perfect balance is impossible due to two factors. First, depending on the connection network for the processors, setup and cleanup processes provide for times when not all the processors are busy. Second, there is usually some fraction of the job that is inherently serial. This part of the job has to be performed on one processor while the other processors are idle. These two factors are innate to the problem. Poor load balancing can also be designed into a problem due to improper decomposition. This balancing can occur in two ways, either static or dynamic.

Dynamic load balancing is adapting the load to the current state of processing. This is a difficult task due to the addition overhead incurred and the meta-level control needed.

Several other performance measurements are needed to baseline a production system. These include 1) the number of productions or rules, 2) the number of working memory elements or facts, 3) the composition of the rules which includes the number of clauses in the LHS and RHS of the rule, and 4) the average number of rules eligible to be selected on a given cycle. These are but a few basic components, other characteristics depend on the system and inference engine being examined.

Since production systems are a type of search, parallel decomposition techniques for search problems can be applied to production systems. So, production systems can be decomposed along the control functions like a branch-and-bound or it can be decomposed by its data. In the case of a production system the data can be thought of as two parts (Figure 2). The first part is the facts or the working memory (WM). Although the working memory can have several meanings in this context it refers to the initial facts and axioms as well as the facts added due to the firing of rules. The second part is the rules or the production memory (PM). These two parts are not always distinct, but can overlap. For example, the result of a rule could be the addition of a new rule. The reasons for making the distinction in the types of data is that in some cases it is much easier to decompose the rules than it is to decompose the facts. The latter requires data dependencies to be worked out while the former requires less restrictive decomposition considerations.

As described earlier the concurrency available in decomposing the functions is limited. This is particular true for production systems where over 90% of the time is spent in the match function (Gupta, 1986). So the main emphasis is placed on the decomposition of the data. The methods for implementing a production system tend to center around ways to decompose the rules (PM) and the facts (WM). This has lead to several algorithms to accomplish this decomposition and their placement on separate processors. Examples include a Full Distribution of Rules, the Original DADO, Miranker's TREAT, and Fine Grain RETE (Shakley, 1987a). These algorithms are generally at the level where the underlying inference engine structure is unimportant. The methods are more concerned with the dependencies of the rules on each other and the facts (WM). The initial prototype knowledge structure consisted of a frame-based data structure of facts and a list structure of rules. The impact of this selection is analyzed in a subsequent section.



Analysis and Design of Inference Engine

The main structure of the piloting control is a layered series of two expert systems. Each expert system has several components organized by functionality (Figure 3). These components provide a source of data independence of rules and working memory. The system contains an "average size" production and working memory. The system contains over 350 rules. The working memory consists of schemata which are frame-like structures. Each frame contains slots that hold the individual facts. There are approximately 160 schemata. The average number of slots per frame is approximately ten, therefore the total number of facts is about five times the number of rules.

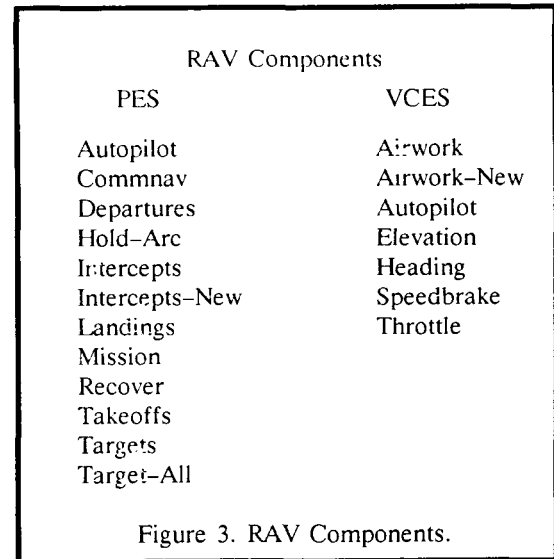
The requirement for an inference engine is to perform the basic production system cycle: match, select, and act. [This cycle is embodied in the resolution process (Nilsson, 1980).] This inference engine software should be able to match the rules of the RAV expert system with the facts in working memory. It should select one of these rules and add the results of the RHS of the selected rule to the working memory.

The current RAV system implemented on the TI Explorer uses the Automated Reasoning Tool (ART) for this process. The lack of availability of ART on the iPSC Hypercube requires that another control process implementation be developed. However, the new implementation should be compatible with the ART rules and working memory structure.

ART is a very complex and extensive tool. To try to rebuild the generic ART system on the hypercube would require a prohibitive development time. Therefore, simplicity of design is a critical component. The new control process should only provide the functionality of ART that the RAV requires.

The previous designed inference engine was modified for parallel implementation. The algorithm for the parallel design is presented in Figure 4. The actual change in the design to the serial inference engine is small. The changes occur in the select and act phase. Each processing element (PE) or node of the parallel design needs to report the rule selection to the system. The PE then has to coordinate with the system in order to act to update the working memory. Three alternatives seem appropriate: a star, a binary tree, or a spanning tree. With the star, one node acts as the central point with all other nodes communicating with that node. This would require longer than nearest neighbor communication or one node hops. The binary tree can be implemented with nearest neighbor communication, but only on higher dimension cubes. The spanning tree (Figure 5) offers the appropriate functionality with nearest neighbor communication. A rule selected on node 14 would be sent to node 6. At node 6, this received rule would be added to the agenda and node 6 would select a rule. This continues up the tree until node 0 receives all the selected rules from its children. It then selects an overall rule and passes it down the tree to all its children. Each child then passes the selected rule to its children until all nodes receive the selected rule for firing. The communication network for the flow of information is designed as a spanning tree. The reason for this type of tree is because it preserves nearest neighbor connections and the height of the tree is the logarithm base 2 of p where p is the number of PEs. A node only communicates with other nodes a distance of one away and the length of a path from the bottom of the tree to the top is the dimension of the cube.

The only design aspect remaining is the methodology of placing rules of the RAV production system components on the parallel system's PEs. The first design to place the rules on the PEs, decomposed the rules by com-



1. Initialize: Place a copy of the simple inference engine on each PE. Place a copy of WM on each PE. Place a subset of the PM on each PE.
2. Repeat until done;
3. Match and select on each PE.
4. Report selection up tree.
5. Overall selection made at root node.
6. Broadcast WM change to all PEs.
7. end repeat;

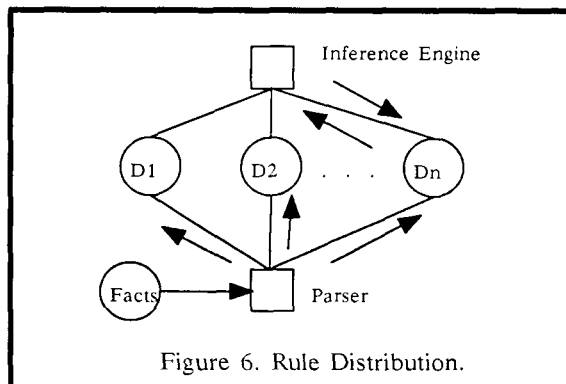
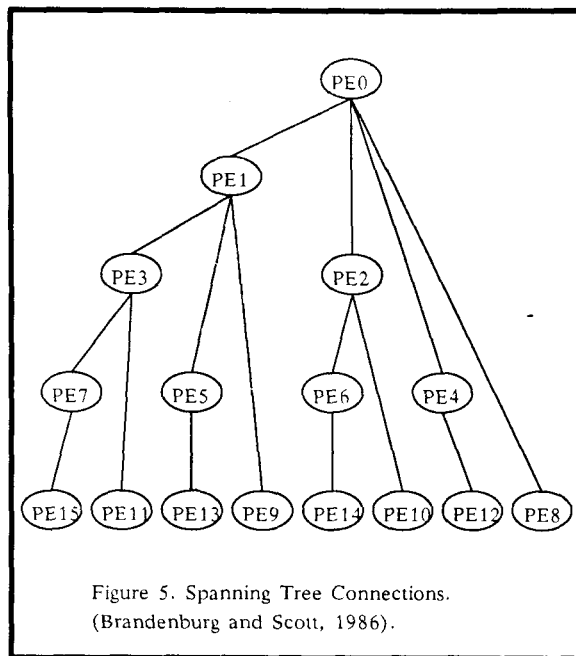
Figure 4. Parallel Inference Engine Algorithm.

ponent on the different PEs. This was unsatisfactory since this produced an uneven load balance. Therefore, a more appropriate decomposition of the rules was to equally distribute the rules to the various PEs (Figure 6).

Low-Level Design & Implementation

The parallel inference engine was implemented in CCLISP on the iPSC Hypercube. Since the serial inference engine was implemented on the TI Explorer using Common Lisp the changes needed due to language differences were minimal. CCLISP (Broekhuysen, 1987b) was not as extensive as Common Lisp on the TI Explorer (Explorer, 1985). For example, CCLISP did not support CAD-DDR, but this was easily changed. The language issues simply did not provide a major obstacle. There was, however, major effort involved in implementing a parallel inference engine. This centered around the communication between nodes.

The parallel design of the inference engine required that the selected rule from all the nodes be collected at one node for the final selection, and then that selection needs to be passed to all the other nodes. This is accomplished with a spanning tree.



The algorithms for determining the parent and children nodes of the tree dependencies uses a logical "or" of the binary node numbers. This was difficult to implement in LISP, so a table look-up was used. This proved to be very simply and efficient, but somewhat inflexible since only node zero can be used as the root node.

The message passing within CCLISP presented some problems. There were several ways to pass messages. They ranged from low level message passing to high level fast loading (FASL) node streams. The low level message passing required that the length of the message length be known. This proved to be a major limitation, given that the messages to be passed would be variable length rules. Therefore, the high level FASL node streams were selected for their abstraction. These streams did provide a problem. There was no defined way to do a receive-wait. This allows a process to enter receive mode until a message is received. This process is very convenient for synchronizing nodes. This function had to be built using a loop doing repeated receives until a message was received from another node. One other note concerning this process. The documented function 'listen' was not implemented (Broekhuysen, 1987b). This would have provided a method to test the message buffer for an incoming message without actually doing a read.

The changes to the actual serial inference engine were small and confined to a small number of modules. The first module had to be changed to provide the proper termination test. This is important to insure that the individual nodes terminated only when no overall rule was available, not just when the node found no matches. The other module had to be changed to incorporate the communications with the other nodes. Several other routines were needed to assist this latter module to make the communication.

The RAV system consisted of the original components of the RAV expert system designed by TI. In its original form it consisted of a series of plans, needs, and schemata which was a higher level abstraction than the ART rules (McNulty, 1987). The plans and needs were then "compiled" into ART rules for execution using software developed by TI (Lystad, 1987). The only way to get the schemata and rules from the RAV Plans and Needs was to compile them into files rather than into the ART system. From there, the rules and schemata are then transformed into a format that the serial and eventually the parallel inference engine could accept. This transformation was partially automated with a routine and further transformed by hand to come up with the final format compatible with the implemented inference engine. The total translation was not done programmatically due to the complexity of the software involved to parse and recognize the various ART syntax forms.

Experimental Analysis of Results

The only test suite available was a demonstration developed by TI midway through the development of the system. In fact, the expert system used in this study was not complete and was only a demonstration prototype (Graham, 1987). This demonstration was considerably lengthy and required the perfect execution of all the rules and implementation of all associated ART functionality. The alternative was to develop small prearranged sets of facts that would trigger a subset of rules. This was the preferable choice since the inference engine could not deal with all the rule formats in their entirety. The complete demonstration was not used, therefore a full comparison of the systems could not be done.

The code and expert system for the parallel RAV system was transported to the iPSC Hypercube from the TI Explorer to a microVAX to a VAX across the Defense Data Network (DDN) to the AFIT VAX and finally to the iPSC Hypercube (Shakley, 1987a). This was perhaps the most "trying" of the problems associated with this whole implementation. This was because of the many machines that had to be traversed to get the code from the TI Explorer to the iPSC. This was only done after the tape-to-tape transfer failed due to a mismatch in tape-formats.

The code was analyzed for its effectiveness and correctness by comparing its execution with that of the ART system. The code performs slowly compared to systems like ART which runs at between 2-30 rules per second (Gupta, 1986). On the Explorer, the inference engine plods along at about one cycle every 30 seconds or 2 rules per minute. This is with the smaller rule base. With the larger rule base, the system runs one cycle every 113 seconds. On the iPSC Hypercube, the serial system runs at one cycle in about 11 seconds with the smaller database and in about 79 seconds with the larger database. This situation involves two phenomena that need explanation. First, why does 60 extra rules slow the process down so much? The reason for this is in the format of these rules.

They are rules that have a variable binding on the schema name. This means they must go through the list of schema names looking for a match. These rules can not take ad-

vantage of the indexing created by the frames. This is a process not handled well by the inference engine. Second, why does the TI Explorer go slower than the iPSC Hypercube? The surface appearance is that the Common Lisp on the Explorer is much more extensive than that on the iPSC Hypercube. These would create more overhead on the Explorer and allow the hypercube to process faster.

The speedups are far less than linear (Figure 7).

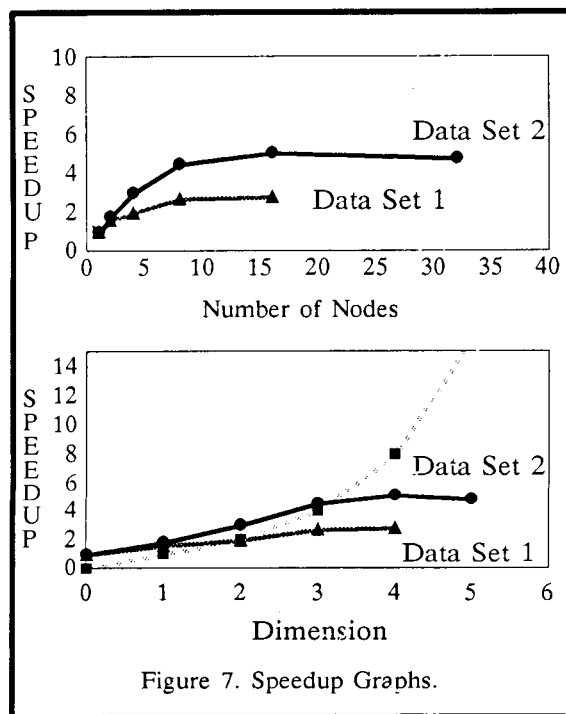


Figure 7. Speedup Graphs.

The speedups taper off with the dimension of the cube. The select time on some of the nodes exceeds the match time on the node. In some instances, the select time far exceeds the match time. There are two explanations for this phenomena. The first is that the problem size is too small for the higher dimension cube. This can be seen from the first data set. On the higher dimension cube, the match time is less than the select time for one series of communication. The result is a much longer cycle than what would be expected from a linear speedup. The second cause for this slowdown is poor load balancing. Each node has an equal number of rules and each rule gets checked for a match on each cycle. The reason the load is imbalanced is due to the composition of the individual rules. The rules have different numbers of clauses that causes each rule to have a variable length match time. Also, the order of the clauses creates different match times. Even though a rule may have a long list of clauses to match, if the first one fails, then the rule matches quickly. Finally, the different types of matches take varying lengths of time depending on the format of the clause. A match involving a schema variable match takes much longer than a simple slot match.

Theoretical Analysis (Shakley, 1987b)

The computational complexity of a production system can be divided into several components. These correspond to the match, select, and act phases. The last component's (act) computational complexity is easiest to analyze. During the act phase, one rule has been selected and

the clauses of the consequent are either added to or retracted from the fact database. Although, it depends on the data structure of the facts, it is a polynomial operation. In most cases, the complexity is either constant or linear. This would occur with any type of indexing or linked list data structure of facts. The computational complexity of the other two components is more difficult to analyze and in fact depends on the types of clauses used within the rules.

A set of parameters is needed to discuss the computational complexity of the production system. These parameters are found in Figure 8.

nR - Number of Rules
 nF - Number of Facts
 IC - Maximum length of a rule clause
 nA - number of antecedent clauses
 nC - number of consequent clauses
 nV - number of variables

Figure 8. Production System Parameters.

The first type of production system considered is one with only constant terms in the rule clauses. The initial observation about the system is that the fact database is unordered. Therefore, the entire database must be searched for each clause of each rule. The worst case time complexity is then $O(nR \cdot nA \cdot nF)$ for the match phase. This complexity grows as the number and complexity of rules grows and as the number of facts grows. In a monotonic system, one in which only facts are added, not retracted, this complexity grows as the system operates adding more facts.

The select phase can take two forms. In the first, the select phase stops whenever the first rule is matched successfully. The second form waits until all the rules have been matched, and then selects one rule from all the successfully matched rules. In the worst case, the matcher matches every rule's clauses to every fact. Therefore, the two forms of the select only affect the complexity of the select. In the first form, the complexity is constant and in the other it is $O(nR)$.

The first form has some disadvantages that make it unattractive. By taking the first rule that matches successfully, the production system is creating an implied order of the rules. This creates preconditions to the subsequent rules. For a rule to be selected, not only does the condition of the rule have to be met, but also the conditions of the previous rules can not be met. Therefore, the second select form creates a certain sense of "randomness" to select process. Therefore the overall complexity of a production system with constant clauses is $O(nR \cdot nA \cdot nF)$ for one match, select, act cycle. It should be noted that the entire production system is NP-Complete and if retractions of facts are allowed the production system is not even guaranteed to terminate.

The second type of rule clause considered is one in which variables are allowed. This type of clause has the potential to match with many different facts. The match complexity of these types of clauses are the same as in the previous case: $O(nR \cdot nA \cdot nF)$. However, the select phase is more complicated. There is a successfully matched rule for each instantiation of each variable. Therefore, the select phase has a worst case time complexity of $O(nR \cdot nV \cdot nF)$. Where the number of variables equal the

number of atomic elements within the clause then nV equals nA times the IC. This produces a complexity of $O(nR*nA*nF*IC)$. Both the preceding cases assume that there is no order to facts. Also, there have been no simplifying assumptions about the structure of the database.

The first step in simplifying the fact data structure is to provide a frame structure for the facts, an Object-Attribute-Value (OAV) structure. This provides a way of ordering the facts. In this way, the value can be accessed by indexing into the fact data structure using the object and the attribute. A clause in a rule can now be matched in constant time, $O(1)$. Therefore, the time to match is reduced to $O(nR*nA)$. The time to select is still $O(nR)$ where no variables are associated with the clauses. If a variable is associated with the value within a clause, then the time to match is the same. However, the time to select is $O(nR*nV)$ or $O(nR*nA)$ since there is only one variable per clause. The problem arises if variables are introduced into either the Object or Attribute fields of the clause. If both all of the OAV are variables then the system degenerates into the case with no ordering of the facts. The time complexity for the system is $O(nR*nA*nF*IC)$. The only case left is if two of the three are variables. The match time $O(nR*nF*nA)$ in the worst case. Each rule must be matched with each fact for each clause in the rule. In conclusion, this simplifying feature alone only save time when used with only one variable item per clause and that item must be the value of the OAV. It should be noted that if the variable is in another field, then the indexing scheme could be changed to account for the change in the clause structure.

From the previous example, it can be seen that better performance can be achieved if the facts can be organized into frame structures that allow for the indexing of facts. This benefit, however, only allows for a variable in the value field. Also, on each cycle this entire match, select, act cycle has to be reaccomplished. A state saving feature within the production system could reduce the time for all but the first match, select, act cycle. This data structure stores from cycle to cycle the rules that had previously matched. Then only those rules that were affected by the selected rule would need be considered on each pass. Therefore, each fact needs an associated list of affected rules. Then, when this fact was changed via a rule only the affected rules would need to be considered for matching. In the worst case, every rule affects every other rule creating the situations in the previous cases with the appropriate time complexities. In this case, this simplification has no improvement. However, rare is the system where every rule affects every other rule. The reduction is actually within the coefficient of the equation. The time complexity for the OAV case is $O(nR*nA)$. The coefficient of this complexity is one. With the above simplification, each subsequent cycle is $(1/aR)*nR*nA$ where aR is the number of affected rules. This method also reduces the cycle time for the multiple variable case of OAV. In the original case, the time complexity was $O(nR*nA*nF)$ for each cycle of the production system. This still holds for the first cycle, but every cycle afterwards is considerably less. In fact, each cycle is $O((1/aR)*nR*nA)$. This is true even for the rules with multiple variable clauses, since a rule can only affect a limited number of instantiations of a multiple variable clause rule. Only when a rule affects all the instantiations of a multiple variable clause rule is the complexity of that following cycle increased to $O(nR*nA*nF)$. The state saving technique does increase the space complexity of the system. Each fact has to have a list of pointers to each rule that is affected by the fact.

What does this all mean in picking an inference data structure and strategy? The answer lies in the problem domain. It revolves around the characteristics of the

system: the number and complexity (number of antecedents) of the rules, and the number and format of the facts. If the format of the facts is "random" in that no ordering can occur, then the system is doomed to poor performance. If on the other hand the facts can be organized into frames or some form of OAV where the value is the only variable field. Then an order of magnitude improvement can be obtained. If variability is allowed in other fields of the frame, then poor performance is reintroduced. If a state saving feature is introduced, then in the average case the performance can be improved on all, but the first cycle of the production system. This state saving feature, however, does require additional space. Therefore, a mixture of inferencing techniques could be the "best" solution.

A study consisted of analyzing the various types of inferencing with the different types of clauses in the Robotic Air Vehicle Expert System. The expert consists of over 300 rules and over 160 frames with approximately 10 slots per frame. The system was originally implemented with an unordered list of facts. The system performed "well" with a subset of rules and facts that did not exceed approximately 10 rules and 50 facts. That is rules that contained only one variable item per clause. One cycle took approximately 1-2 minutes. The system was then upgraded to a frame based data structure [In the first case, each slot was transformed into a single fact and treated as unordered]. This system performed "well" with all rules and frames. The system completed a cycle in approximately 30 seconds. However, this system had problems with clauses that had multiple variable items. These rules took approximately 1 minute per rule. The cycle time was never calculated due to the extremely long time per rule. It should be noted that the rules of this type were limited in numbers and were less than 50. The next step was to create a state saving system. This system only matched and considered the rules that were affected by the previously selected rule. This required that the rules be compiled into the data structure of the facts, so that each fact could point to the rules that were affected by the modification of that fact. The first cycle of this system still took approximately 30 seconds to complete without the multiple variable items. Then each subsequent cycle took approximately 5-10 seconds thereafter. The sequence of different inferencing techniques confirmed to the prediction of the computational complexity analysis. Further, work is still desirable to provide an even broader base of sample data. This would help to confirm even further the time complexity analysis.

Conclusions

This research study investigated the feasibility of parallel architectures to improve the performance of potential real-time software. In particular, the feasibility of parallel architectures to improve the NP-complete problem of state space search in the form of a production system. The RAV expert system is an example of such a system.

The speedup results from the inference engine were disappointing, however, it did show that speedups were possible. The speedups suffered from a combination of two factors. The first was a relatively small problem compared to the communications overhead. It was observed that for a system with greater than eight nodes, the time to perform the match cycle on a node was less than the time to communicate with the other nodes. Also, the speedup suffered due to a load imbalance. The method for decomposing the rules proved to be unsatisfactory. The method did not take into account the variability among rules. Real-time performance was not achieved as was anticipated. With improvements in the inference engine and

load balance, significant improvements could be possible.

The performance of the iPSC Hypercube in comparison to the TI Explorer was fairly positive. The iPSC Hypercube performed about twice as fast as the TI Explorer on the inference engine. However, this seems to be due to the simplicity of the LISP on the iPSC Hypercube.

The inference engine developed in this study performed adequately. The inference engine fired a rule about once every 30 seconds or at a rate of just under 2 a minute on the TI Explorer. The engine fired a rule one every 18 seconds on a single node of the iPSC Hypercube or just over 3 rules a minute.

Recommendations

This study probably raises more questions than it answers. Beginning with the serial inference engine. An area of study would be the performance of inference engines. The characteristics of inference engines and their performance would have been invaluable to this research. More work could be done to improve the inference engine in this study. The inference engine in this study could be redone using the Rete algorithm. The benchmarking of inference engines and inference engine techniques would be valuable. Also, with regard to inference engines, this study started to automate and simulate the functionality of ART on the iPSC Hypercube. The further development of the process could provide a valuable tool for expert system development. The expert system could be developed on the TI Explorer using ART and transferred to the iPSC Hypercube for performance studies.

There needs to be better ways to characterize the work needed to match a rule so that more effective iPSC Hypercube load balancing can be performed (Bailor and Seward, 1988). This would depend on the structure of the inference engine, the structure of the rules and the structure of the facts for efficient partitioning and distribution of data. An automated means should be developed to handle this load balancing. This automated method could be used for adaptive time-based load balancing. In addition, temporal dependency mechanisms could be used to achieve this adaptive load balance.

The current study used a complete set of facts on each node and only one rule was fired at a time across the entire network. The RAV software showed promise for further levels of concurrency. The structure of the RAV expert system shows great promise in firing several rules in one cycle of the inference engine. This could provide a significant time and space savings. The RAV expert system is to operate in a real-time environment. This means a varying time requirement for operations. The implementation of an automatic way to dynamically increase the speed of a computation through a meta-level of knowledge and control would be valuable to the design of real-time software.

Bibliography

- Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley Publishing Company, 1974.
- Bailor, Paul D. and Walter D. Seward. "A Generalized Strategy for Partitioning and Distributing Data in Data Parallel Algorithms." Unpublished report. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, June 1988.
- Barr, Avron and Edward A. Feigenbaum. *The Handbook of Artificial Intelligence, Volume 1*. Stanford, CA: HeurisTech Press, 1981.
- Brandenburg, Joseph E. and David S. Scott. *Embeddings of Communication Trees and Grids into Hypercubes*. iPSC Technical Report No 1. Intel Scientific Computers, Beaverton, OR, 1986.
- Broekhuysen, Martin, editor. *Concurrent Common Lisp User's Guide Version 1.1*. Cambridge, MA: Gold Hill Computers, 1987a.
- , *Concurrent Common Lisp Reference Manual Version 1.1*. Cambridge, MA: Gold Hill Computers, 1987b.
- Explorer. Lisp Reference (2243201-0001). Texas Instruments Incorporated, Austin, TX, June 1985.
- Fischler, Martin A. and Oscar Firschein. *Intelligence: The Eye, the Brain, and the Computer*. Reading, MA: Addison-Wesley Publishing Company, 1987.
- Gupta, Anoop. *Parallelism in Production Systems*. PhD dissertation. Carnegie-Mellon University, Pittsburgh, PA, March 1986.
- Ishida, Toru and Salvatore J. Stolfo. "Towards the Parallel Execution of Rules in Production System Programs," *Proceedings of the International Conference on Parallel Processing* 568-575 (1985).
- Kornfeld, William A. "The Use of Parallelism to Implement a Heuristic Search," *Proceedings of the 7th International Joint Conference on Artificial Intelligence* 575-580 (1981).
- Lystad, Garr S. "The TI Dallas Inference Engine (TIDIE) Knowledge Representation System," *Proceeding of the IEEE National Aerospace and Electronics Conference* 1348-1351 (May 1987).
- McNulty, Christa. "Knowledge Engineering for Piloting Expert System," *Proceedings of the IEEE National Aerospace and Electronics Conference* 1326-1330 (May 1987).
- Nilsson, N. J. *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga Publishing Company, 1980.
- Norman, Capt Douglas O. *Reasoning in Real-Time for the Pilot Associate: An Examination of Model Based Approach to Reasoning in Real-Time for Artificial Intelligence Systems using a Distributed Architecture*, MS Thesis AFIT/GC/ENG/85D-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1985.
- Shakley, Donald J. *Parallel Artificial Intelligence Search Techniques for Real-Time Applications*, MS Thesis AFIT/GC/ENG/87D-24. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987a.
- , *Order-of Analysis of Production Systems*. Unpublished report. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987b.
- Ward, Paul T. and Stephen J. Mellor. *Structured Development for Real-Time Systems Volume 1: Introduction and Tools*. New York, NY: Yourdon Press, 1985.

PORTABLE INFERENCE ENGINE: An Extended CLIPS for Real-Time Production Systems

Thach Le & Peter Homeier
The Aerospace Corporation

Abstract

The present CLIPS (C-Language Integrated Production System) architecture has not been optimized to deal with the constraints of real-time production systems. Matching in CLIPS is based on the Rete Net algorithm, whose assumption of working memory stability might fail to be satisfied in a system subject to real-time dataflow. Further, the CLIPS forward-chaining control mechanism with a predefined conflict resolution strategy may not effectively focus the system's attention on situation-dependent current priorities, or appropriately address different kinds of knowledge which might appear in a given application. Portable Inference Engine (PIE) is a production system architecture based on CLIPS which attempts to create a more general tool while addressing the problems of real-time expert systems. Features of the PIE design include a modular knowledge base, a modified Rete Net algorithm, a bi-directional control strategy, and multiple user-defined conflict resolution strategies. This paper will analyze the problems associated with real-time applications, and explain how the PIE architecture addresses these problems.

1. Introduction

Expert system technology has been successfully applied to a number of practical applications [1]. As this technology becomes more widely used and starts to address more complicated real world problems, issues of portability and accommodation of real-time data will become more significant.

The ability to handle real-time data will automate processes and remove the human dependency for some cognitive preprocessing of inputs. For example, a real-time satellite diagnosis expert system, instead of asking the operator to analyze satellite telemetry data, will take in the raw telemetry data directly and extract the needed information. This capability is important, especially in a situation where the amount of real-time data precludes exhaustive human preprocessing. This situation exists in many aerospace applications.

CLIPS, an OPS5-like (Official Production System) production system written in the C language by NASA, is an excellent delivery vehicle because of its portability. However, it was not designed for real-time systems. PIE (Portable Inference Engine) is a production system based on CLIPS which attempts to avoid degradation in performance when the system is subject to real-time dataflow and also supports a more explicit control mechanism.

In part two and three of this paper, we will briefly review a typical production architecture as exemplified by CLIPS and discuss the problems which arise in real-time systems when employing this type of architecture. In part four and five, we describe the PIE architecture, design issues, and how this architecture accommodates these problems. PIE is an on-going project, and its architecture is currently being implemented for embedded applications.

2. CLIPS & Production Systems Architecture

A production system is a typical rule-based system. It consists of a knowledge base and an inference engine. The problem state, contained in a data structure called working memory, is represented by facts. These facts may be created during the problem-solving process, either through rule execution or via the external environment. The knowledge base, which resides in a data structure called production memory, contains rules of the form "IF antecedent THEN consequent". Usually the antecedent is a set of patterns representing the rule's conditions, and the consequent represents conclusions or actions.

Production systems may be data-driven, goal-driven, or some combination of the two. CLIPS' behavior rests on the frequent re-evaluation of the problem state (represented by the current memory elements in the working memory), rather than on any static control structure of the program. Therefore, CLIPS falls into the class of data-driven production systems.

A production system is usually described in terms of recognize/act cycles, which may be divided into three separate processes:

1. Matching: Match a set of existing facts in working memory, which represents the current problem state, against all available rules. Rules whose conditions are satisfied are called instantiated. The set of instantiated rules at any cycle is called the conflict set. The matching process updates the conflict set at each cycle.

2. Conflict resolution: Select a single rule from the conflict set based on some criteria, which could be (as in the CLIPS case) user-predefined priority of rules, the recency of working memory elements, the number and complexity in rules' patterns.

3. Execution: Carry out the actions specified on the Right Hand Side of the selected rule. This could affect the content of the working memory (change the problem state).

The production system performs this recognize/act cycle

repeatedly until encountering an empty conflict set or a halt action. The production system as a model of computation provides a powerful context within which large, ill-structured problems may be described [2].

3. Real-time Issues

The term "real-time" is not easily defined. It is usually associated with fast response. A more precise definition is a system which has guaranteed response time for a defined class of events. In general, the design of a real-time system involves an integrated hardware/software approach, with careful prioritization of competing service requests. A more limited definition for real-time is adopted here. For production systems, we define a real-time system as one where efficiency is a primary design concern, which allows the generation of working memory elements from on-line inputs, and which has the capability for conflict resolution based on dynamic prioritization.

In the following sections, the real-time issues that are related to the performance of CLIPS are discussed. When the system is subject to real-time dataflow, two main sources of deficiency are identified: the control mechanism and the matching process.

3.1 Control

CLIPS provides an implicit control mechanism, built-in to the production system to govern the direction of the inference engine. This control algorithm is specified in the conflict resolution strategy. The CLIPS conflict resolution approach is to select a rule from the conflict set according to its priority, the recency of the working memory elements that match the rule conditions, and the specificity of the conditions (measured by the number of tests performed). In addition, rules that have previously fired will not be fired again on the same facts or working memory elements.

To exploit the full power of the CLIPS language, the application programs should be data-driven. The course of execution, or the sequence of rule firings, should be sensitive to the characteristics of the data. Such systems, where the direction of problem-solving is from facts toward goals, are characterized as having a forward-chaining control strategy. Other systems might use a backward-chaining, or goal-driven, control strategy, where the direction is from goals toward facts.

For a given search space, the best direction of reasoning is to move in the direction of less alternatives to minimize backtracking, a process of returning to the parent search node to explore other alternatives if the current search node is not satisfied [4]. Hence, the decision to use either forward-chaining or backward-chaining is dependent on the structure of the search space [5]. However, many realistic problems do not have a simply structured search space. For such problems, an efficient reasoning strategy will be bi-directional (i.e., combining both backward- and forward-chaining).

A bi-directional control strategy, beside enhancing the searching process performance, is also useful in expressing the way of human experts do problem solving [3]. He or she often alters the line of reasoning and sets up different hypotheses, or goals, if a particular fact is observed.

A problem with a predefined conflict resolution strategy is that its algorithm is not appropriate for all applications. Different applications or different parts of the same

application might require different control knowledge specific to their domains. For example, many applications select rules based on confidence factors, or perhaps some rule-of-thumb provided by an expert. Such knowledge has to be embedded within rules, whereas the appropriate place is the conflict resolution. The code as a result will be harder to understand. There are also performance penalties. The system, instead of taking one recognize/act cycle to select the right rule, has to perform a few recognize/act cycles to come up with the same result. In time-critical situations, explicit control may be needed to quickly resolve conflict.

In building a large expert system, one may encounter many problems that require various control techniques to keep the system's performance efficient. As a real-time production system language, PIE must fulfil this requirement.

3.2 Matching

Matching in production systems is the process of collecting a set of rules, the so called conflict set, that have their conditions satisfied by the current problem state. This process takes about 90% of a recognize/act cycle [6]; therefore, the performance of the matching algorithm is crucial to the overall performance of expert systems.

CLIPS uses the Rete algorithm for matching. Rete is known to be the most efficient matching algorithm for many patterns to many objects [7]. It is implemented in several popular expert system shells [8]. One major assumption that contributes to Rete's efficiency is the assumption that the problem state changes slowly. This assumption is valid for many systems where the problem state is changed only by the execution of rules. Since each rule is usually small, its effects on the problem state should also be small. The Rete match algorithm capitalizes on this observation by saving the previous cycle's matching information, and only updating the matches that have changed. However, in a real-time situation, the problem state could be changed, possibly massively, by the external environment. This violates the Rete Net's assumption, and Rete should not be used in such cases [9].

As mentioned before, some degree of backward chaining is often incorporated into a forward-chaining system. In this case, the resulting system behaves as if it only focuses its attention on the most recent goal; but in reality, at the underlying level, the system still spends its computational resources matching all rules, which includes many that are not relevant to the current goal. For example, if the current task of an automobile diagnosis system is to check out a problem with the battery, it should not devote much of its computing resources to match rules related to incoming data from the engine.

4. PIE

PIE is a production system based on CLIPS with architectural modifications to increase applicability to real-time systems and to provide enhanced production system capabilities.

4.1 Design Issues

As seen above, the potential unsuitability of CLIPS as a language for building real-time expert systems results from the built-in control mechanism, and from assumptions of the Rete matching algorithm. In order to overcome these

shortcomings, PIE has two requirements. First, the matching algorithm should be sensitive to the changed data that are relevant to the current task being solved, but not to the total amount of changed data. Second, the control mechanism has to be more explicit and flexible enough to accommodate different kinds of knowledge which might appear in a single application.

The primary architectural features of PIE which differ from CLIPS are a modular knowledge base, a modified Rete match algorithm, a bi-directional control strategy, and multiple user-defined conflict resolution strategies. The following section will detail these features.

4.2 Architecture

Beside rules, PIE has two other data structures, called modules and goals. A module is a set of rules that are grouped together based on their functionality or any other convenient criteria that the programmer defines. A goal is a set of modules that defines a particular task to be achieved. A module could belong to more than one goal; in this way, multiple definition of a set of rules that belong to more than one goal can be avoided.

At any time, there will only be one active goal. Only the active goal's rules are to be considered in the current recognize/act cycle. Corresponding to this set of active rules is a set of active working memory elements which are defined in the active rules' conditions. Matching has to be done between the active rules and the active working memory elements only; other rules and working memory elements will be ignored as long as they remain passive.

A rule, once selected to fire, could activate a new goal or deactivate the current goal. Initially, a top-level goal is activated. If a new goal X is activated by a rule in the top-level, then X is said to be a child of the top-level goal, and the top-level goal is said to be a parent of X. When activated, a goal's rules become active and its parent's rules become passive. When deactivated, a goal's parent rules will become active.

Associated with each module is a conflict resolution strategy that can be defined either by the programmer or by a default strategy. Modules included in the same goal must share the same conflict resolution strategy. The conflict resolution is defined in a procedural language (C), and can access system information such as number of instantiations, rule priority, recency factor, number of tests, condition patterns of rules, etc.

The system at the highest level is a goal tree that behaves like a goal-driven/backward-chaining system. At the goal level, it behaves like a purely data-driven/forward-chaining system with a user-defined conflict resolution strategy. With this architecture, it is possible to build any level of integration of forward-chaining and backward-chaining (See figure 1). For example, a purely forward-chaining system, such as CLIPS, is a PIE with a single goal. A purely backward-chaining system is a goal tree in which each goal contains rules to invoke subgoals, except for the leaf goals of the goal tree which contains rules that match to facts.

The Rete Match algorithm compiles all rules in production memory into a dataflow graph called the Rete Net [7]. The matching information is saved in each node of the graph. For

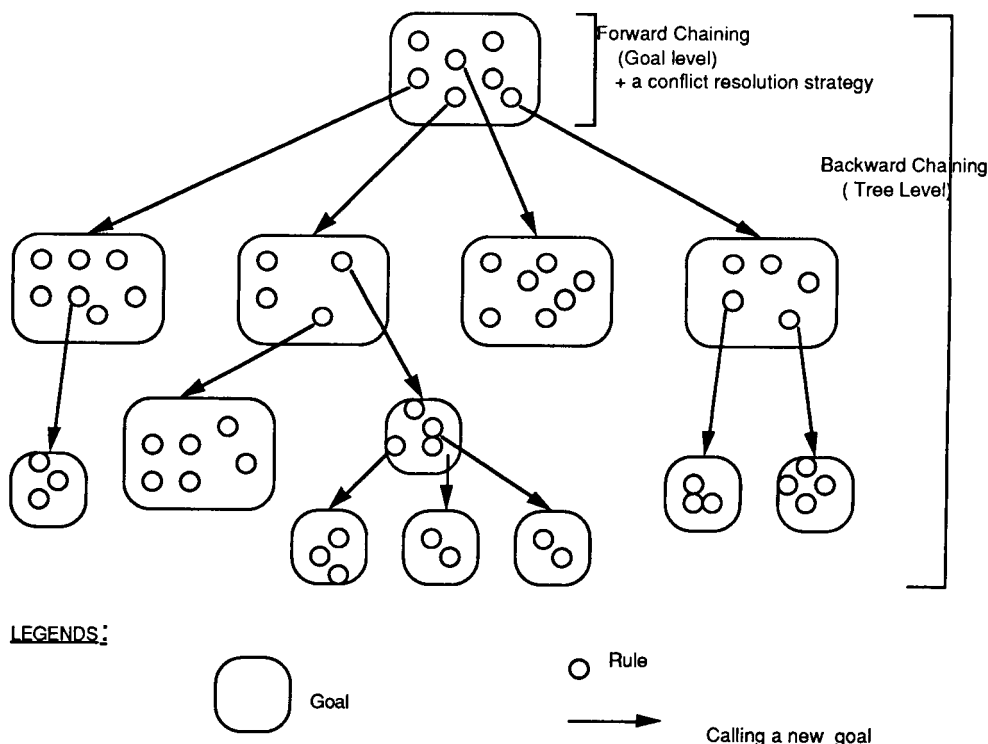


Figure 1: Forward Chaining and Backward Chaining in PIE

ORIGINAL PAGE IS OF POOR QUALITY

PIE, Rete is modified so that at each recognize/act cycle, it only updates a partial net related to rules belonging to the current active goal, instead of updating the whole net. A controller is added to the Rete algorithm to keep track of the relationship among rules, modules, and goals. During the course of execution there usually are many goals to be activated and deactivated. The controller's function is to turn on the appropriate nodes to be involved in the matching for the current active goal and turn off those of the deactivated goal (see figure 2).

The incoming data of an inactive goal will overwrite the older

data but will not participate in the matching process until the goal is activated. If older data are to be saved in time order for some later use, the corresponding patterns created have to include the time index defined by programmers.

PIE is an on-going project. The major effort has been to understand the implementation of Rete algorithm in CLIPS and to build a controller for Rete. The user-defined conflict resolution strategies written in the C language will be compiled and called appropriately with the activation of corresponding goals. Other work will be to enhance the parser to recognize modules, goals, activation, deactivation, etc.

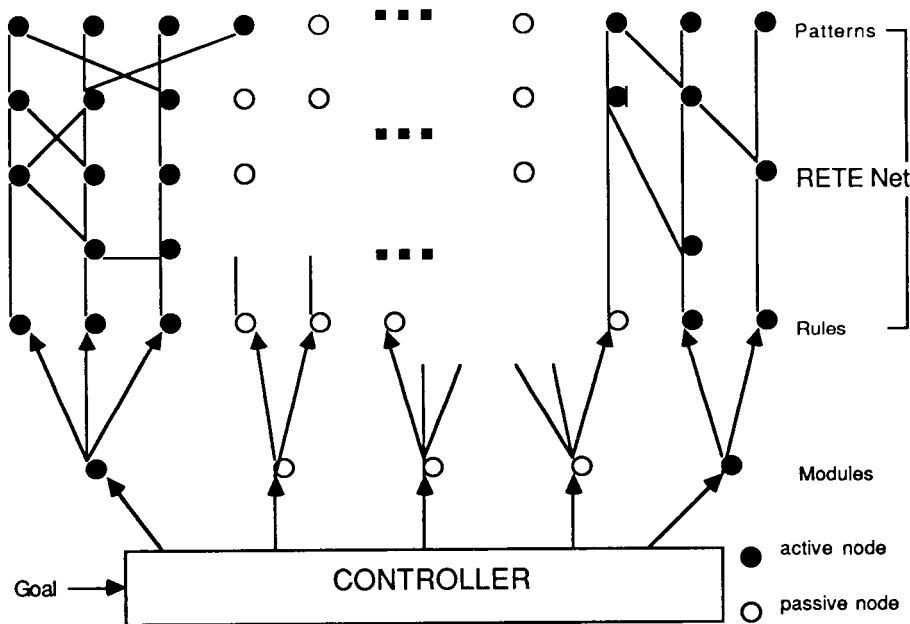


Figure 2: Controller For Rete Match Algorithm

5. Advantages of the PIE Architecture

As stated previously, modularity and the concept of an active rule set decreases the magnitude of the matching problem, permitting use of the Rete match algorithm in a rapidly changing problem state. Each goal of a goal tree might associate with a number of on-line inputs, so the cost of matching for each goal over a recognize/act cycle is proportional to this number of inputs. During the course of execution, the system might explore only part of the goal tree before the solution is found; by focusing the matching process on the currently active goal, the system computing resources are not spent on rule matching for goals never explored.

In addition, supporting modularity at the language level strips away much of the bookkeeping usually needed at the programming level to achieve the same purpose. The result is clearer, easier to understand programs, assisting debugging and maintenance.

Large expert systems also benefit from modularity. Because of the recursive nature of PIE, where each goal can be thought as

a PIE system by itself, a problem can be broken down into subproblems or goals, then can be tackled independently. On the other hand, one PIE system can be integrated into another PIE system as a new goal at any appropriate level of the goal tree.

The integrated forward- and backward-chaining control strategy of PIE optimizes the search process of many realistic problems whose search space does not directly support either a simple forward-chaining or a simple backward-chaining strategy. Simulated backward-chaining is no longer required. A programmer can look at the search space structure of his or her problem and decide on the appropriate strategies for different portions of the search space.

The computational cost of a production system is due to two elements: the rule application cost and the control costs. In figure 3, the cost of rule application is high if the level of "informedness" (i.e., encoded knowledge) of the control strategy is low, and vice versa [4]. With the availability of user-defined conflict resolution strategies, programmers determine the right level of informedness for the control strategy. This will optimize the overall computational cost of

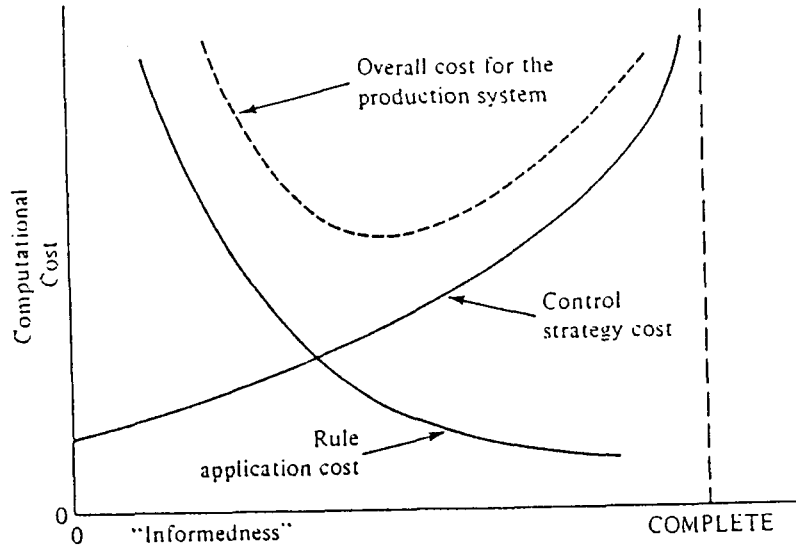


Figure 3: Computational Cost of AI Production Systems [4].

the production system while reducing the obscurity of control knowledge embedded in the rule set. User-defined conflict resolution also helps to preserve the data-driven or the goal-driven nature of rules.

From the design of PIE architecture, the predicted performance of PIE and of CLIPS as functions of the amount of incoming data is shown in the figure 4. If there is no incoming external data, the performance of PIE is likely to be lower

than CLIPS due to additional overhead. As the amount of incoming data increases, CLIPS performance will degrade because of the increasingly invalid assumption used in the Rete matching algorithm. PIE performance is expected to degrade more slowly because of its insensitivity to the total amount of incoming data. The degradation of PIE performance depends mainly on the amount of incoming data associated with activated goals. The programmers have the flexibility to improve the performance by designing the system so that this number can be reduced to as much as possible.

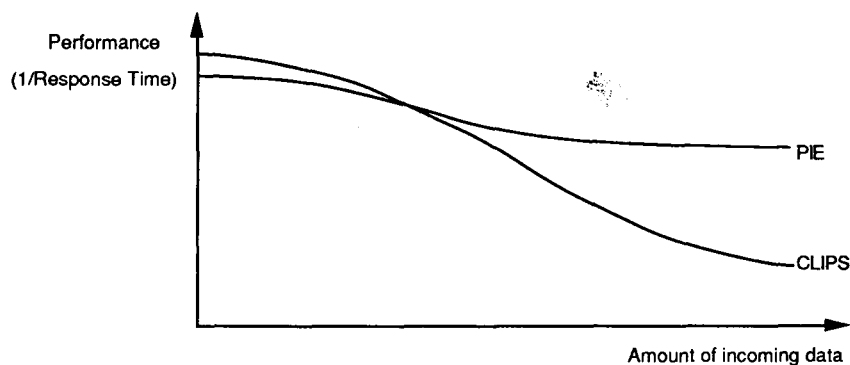


Figure 4: The predicted relative performance of PIE and CLIPS vs the amount of incoming data

6. Conclusion

A PIE architecture has been defined based on extending CLIPS to accommodate modular, hierarchical data structures. The concept of an active rule set and associated working memory subset was introduced to "focus the production system attention". The resultant architecture provides programmers with more flexibility in defining the control strategy of the inference engine. This structure appears promising for systems with real-time constraints, and provides a clear delineation of the control structure from the rule base. This architecture is currently being implemented for embedded applications.

Some applications might have requirements that exceed the capability of PIE in a sequential processing environment. Performance could be improved by use of parallel computing. PIE modularity and independence of control for each goal seem to lend PIE to a macro-level of parallelism, permitting partitioning and implementation on a parallel architecture.

Other areas of interest include extension to temporal reasoning [10], debugging environments for PIE-like architectures, and integration of the knowledge base with the more conventional database structures.

Acknowledgements

The authors would like to thank Dr. Charlie Crummer, Dr. Russ Abbott, and Dr. Jim Hamilton who made many valuable comments towards this paper.

References

1. Michie, D., "Expert Systems," *The Computer Journal*, vol. 23, 1980, p. 369-376.
2. Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5*, Addison Wesley Publishing Company, 1985.
3. Geogeff, M. and Bonollo, U., "Procedural Production Systems," *Proceeding of the Eighth International Joint Conference on Artificial Intelligence*, vol. 1, 8-12 Aug 1983, Karlsruhe, West Germany, p. 151-157.
4. Nilsson, N., *Principles of Artificial Intelligence*, Tioga Publishing Company, Palo Alto, California, 1980.
5. Winston, P., *Artificial Intelligence*, Second Edition, Addison Wesley Publishing Company, Jul 1984.
6. Gupta A., "Parallelism in Production Systems: The Sources and the Expected Speed Up in Expert Systems and Their Applications," *Fifth International Workshop Agence de l'Informatique*, Avignon, France, 1985, p. 26-57.
7. Forgy, C. and Shepard, S., "Rete: a Fast Match Algorithm," *AI Expert*, Jan 1987, p. 34-40.
8. Mettry, W., "An Assessment of Tools for Build Large Knowledge-Base Systems", *AI Magazine*, Winter 1987, p. 81-89.
9. Forgy, C., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, 1982, p. 17-37.
10. Laffey, T., Cox, P., Schmidt, J., Kao, S., and Read, J., "Real-Time Knowledge-Based Systems," *AI Magazine*, Spring 1988, p. 27-45.

ORIGINAL PAGE IS
OF POOR QUALITY

BROWSING SCHEMATICS: QUERY-FILTERED GRAPHS WITH CONTEXT NODES

Eugene C. Ciccarelli
Bonnie A. Nardi

IntelliCorp
1975 El Camino Real West
Mountain View, California 94040

Abstract

This paper reports the early results of a research project to create tools for building interfaces to intelligent systems on the NASA Space Station. One such tool is the Schematic Browser which helps users engaged in engineering problem solving find and select schematics from among a large set. Users query for schematics with certain components, and the Schematic Browser presents a graph whose nodes represent the schematics with those components. The query greatly reduces the number of choices presented to the user, filtering the graph to a manageable size. Users can reformulate and refine the query serially until they locate the schematics of interest. To help users maintain orientation as they navigate a large body of data, the graph also includes nodes which are not matches but provide global and local context for the matching nodes. Context nodes include landmarks, ancestors, siblings, children and previous matches.

INTRODUCTION

As part of the Space Station effort, The National Aeronautics and Space Administration (NASA) is sponsoring several demonstration projects to show the utility of intelligent systems built with knowledge-based software technology. The Space Station will be an extremely complex system composed of many subsystems, each with considerable complexity of its own. Managing the complexity inherent in the structure and behavior of the engineered systems is a major challenge. The use of intelligent systems is intended to help manage that complexity by automating some of the functionality that is currently handled for analogous systems in a manual mode. But it will be critical that humans be able to understand, interact with, and control the intelligent system when necessary. To that end, suitable human-computer interfaces must be built.

Our project, funded by the NASA Johnson Space Center, is to build some tools for the

construction of those interfaces. This paper reports the early results of our efforts to build one such tool, a graphic browser for retrieving schematics (engineering diagrams). The general context of use for the browser in our project is a diagnostic task, although it would be useful for other tasks such as training and maintenance. End users of the interfaces built with the tools are flight control engineers, test engineers and astronauts. Users of the tools are knowledge system developers. We are using the thermal bus (heat transport system) of the Space Station as an example of an engineered system for demonstrating our work.

The use of schematics is central to engineering problem solving. Schematics show connectivity between components in an engineered system. Understanding connectivity is critical to understanding system function and having the ability to detect and diagnose problems. Schematics provide simple, clear models of the systems they represent, allowing the problem

solver to envision connections and trace out and isolate components with faults.

For a large engineered system, there may be thousands of schematics. Rasmussen [1], in his book on interfaces for computer models of engineered systems, observed that coping with the complexity of these systems requires "a large repertoire of different mental representations of the environment." Davis [2], in an article on the use of intelligent systems on the Space Station, noted that good engineering depends in part on the judicious selection of the correct model for a given problem. In a real-time engineering context, not to mention the fragile environment of space, the user needs to have the best possible representation of the problem for maximum problem solving effectiveness.

Large numbers of schematics accumulate to depict an engineered system because engineers must be able to look at the system at different levels of detail, and from different perspectives. NASA engineers described to us four levels of schematic detail: the overview of the complete system, a view of each major subsystem, a view of a component area within a subsystem, and a detailed view of a single component. As one engineer put it, engineers need "information all the way down". Perspectives for a thermal system include mechanical, electrical and thermal perspectives. As problem solution proceeds, engineers move among the schematics to find the view that best captures that aspect of the problem they are working on at a given moment.

Because the Space Station is still in the design stages, we don't know how many schematics will describe it or its component subsystems such as the thermal bus. A comparable number for a nuclear power plant is about 2000 schematics in the "working set" (although the total number of

schematics for the plant may be much higher) [3]. For a particular problem then, finding the right schematic out of a large universe is an important aspect of solving the problem.

INTERFACE PRECEPTS

One of the challenging things about building interface tools is figuring out what will ultimately help the end user of the interfaces developers build with the tools. Tools have a life of their own beyond the fact that they merely make it easier to construct something; more importantly, *tools define possibilities for what to build*. If developers have a tool for constructing cascading menus for example, suddenly we will see cascading menus appearing in our interfaces (for better or for worse). When tools suggest a good representation and provide useful functionality, they can be very powerful.

To maximize the problem solving effectiveness of our users as they work with large quantities of data, we are most concerned with providing tools that enable developers to build interfaces that help the end user to:

- Preserve focus
- Maintain orientation.

Focus

As much as possible, users should be able to devote their attention to solving the problem at hand, not managing the computer. They should, in other words, be able to focus directly on getting their work done, with a minimal allocation of mental energy for interpreting information or taking action that does not bear directly on the problem solution.

The interface should be constructed so that it presents only the data of interest or relevance to the user for the phase of the task in progress at a given moment. The reduction of distracting

material has an important effect on the user's efficiency [4] and irrelevant information should be removed to reduce the effort of ignoring it [5].

Orientation

In any system where users work with a large amount of data, it is easy to become disoriented, that is to lose a sense of where one is in the local or global environment [6], [7], [8]. In the world of the small, two-dimensional computer screen, the spatial cues used for orientation in everyday life no longer apply, or apply in extremely limited ways [9], [10], [11], [12]. The interface must introduce other conventions for orienting the user.

The orientation problem is of course a central concern for browsers, whose very purpose is to navigate around a large space. Users need to know where they are so that they can:

- Move to other places in the data space
- Interpret the meaning of unfamiliar information.

The possible meaning of say, an unfamiliar node on a graph, may be discerned by inspecting the nodes which are nearby and are likely to be similar, and by seeing where the node is in the total graph.

Also, to help users keep a sense of "place", an interface should present an aspect of stability in what is viewed. It is important to minimize the time the user spends adjusting to changing views, unless those views directly reflect some aspect of the problem data themselves.

THE SCHEMATIC BROWSER

We deem a browsing tool an important part of an interface toolkit for our developers whose end users routinely deal with large quantities of data. It is not always possible for users to know

at the outset of a problem solving session exactly what they are looking for. Having the ability to browse quickly and easily should enable users to get to the schematics which best represent their problems.

Browsing is a visual activity. The user sees a set of choices, in some format, from which to select one to view in more detail. An unordered list exemplifies perhaps the simplest representation of choices. With this representation, however, lie two problems.

- There may be too many choices to look at.
- It may be too confusing to easily recognize a desirable choice.

One solution to the recognition problem takes advantage of structure inherent in the set of choices by relating them to one another in some way. Relations may be expressed, for example, in an ordered list, or a graph, as in Figure One.

The graph for the schematics in our application is a loosely-defined part-whole graph. Each node represents an actual schematic, and its children are schematics that show some part of the parent node's schematic in more detail. For example, the "Thermal Bus Overview" schematic (Figure Two) shows the three major parts of the thermal bus, viz., the evaporator (heat acquisition), transport and condenser (heat rejection) sections. The child nodes of the "Thermal Bus Overview" node on the graph (Figure 1) are "Evaporator Section", "Transport Section" and "Condenser Section". Each of the schematics for "Evaporator Section", "Transport Section" and "Condenser Section" shows its section in more detail than can be found in the "Thermal Bus Overview" schematic.

In looking at the graph, the user views nodes

ORIGINAL PAGE IS
OF POOR QUALITY

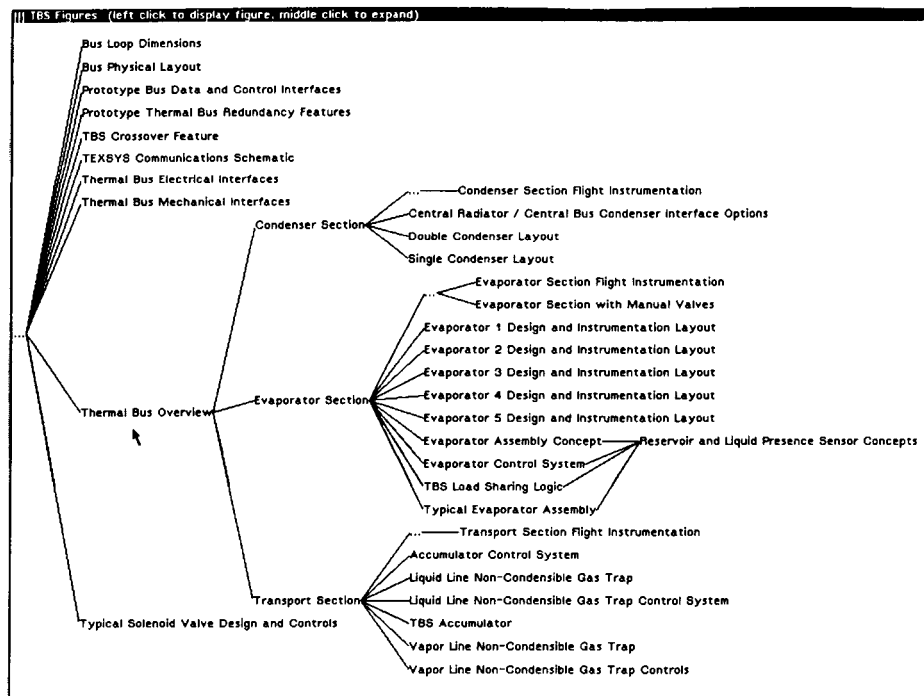


Figure One. Unfiltered part-whole graph of schematic diagrams. User selects a schematic for display by clicking on a node. The unfiltered graph shows too many choices for easy browsing.

ORIGINAL PAGE IS
OF POOR QUALITY

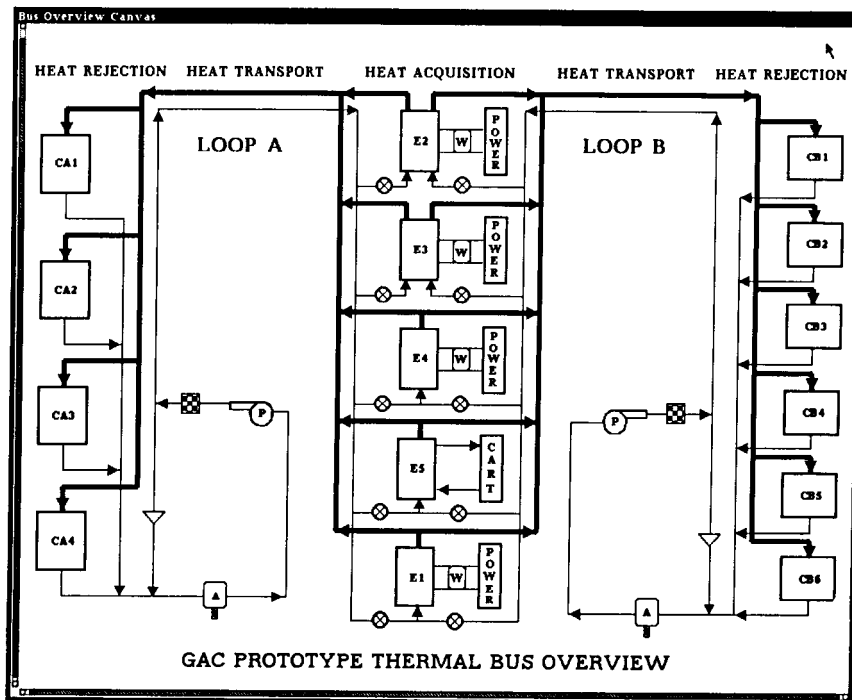


Figure Two. Thermal Bus Overview Schematic. A sample schematic being browsed. This schematic shows the major subsystems of the thermal bus: the condenser, transport and evaporator sections. Each subsystem has its own schematic in which it is shown in greater detail. In the graph, the subsystem schematics are children of the "Thermal Bus Overview" schematic.

which appear in the context of related nodes. The user finds significant orientation in the relations among the nodes so that interesting or relevant nodes are more apparent.

But too many choices remain. The graph lacks sufficient focus. Figure One for example has 36 choices, possibly too many to browse quickly and easily, and the number of choices in an application may of course be much greater. How can the interface filter the choices in an intelligent way and present the choices to their best advantage?

There is a natural tension between the need to filter to promote focus, and the need to provide context for orientation. To study this problem, much of our attention has turned to filtering the graph to a more manageable size, while at the same time providing sufficient context for the interpretation of the nodes.

Our general approach is to:

1. Filter out as many nodes as as possible
2. Add back in selected nodes for context.

The Query-Filtered Graph

First, the user or the intelligent system formulates a query to filter the set of choices and its graph down to a much smaller -- and more relevant -- subset. The query can be thought of as mapping a matching-predicate over the objects in the total set and collecting those that satisfy the predicate. For example, in Figure Three, the user has asked for a graph of schematics that show Evaporator-1. All of the nodes for the schematics containing Evaporator-1, that is, those that match the query, are shown.

In addition, the user sees two nodes that do not actually show Evaporator-1: "Condenser Section" and "Transport Section". As major

subsystems of the thermal bus, the "Condenser Section" and "Transport Section" are included as *landmarks* in the graph, providing recognizable geography or shape to the graph. Landmarks are static nodes, appearing in every graph, to provide a sense of large-scale structure. The user therefore views a graph of the matches for the query without losing a sense of the overall shape of the graph.

Non-matching ancestor nodes are also added back into the graph, again to provide the large-scale, global structural context for the matches. Ancestors provide shape and major reference points in the graph. In Figure Three, we do not see non-matching ancestors as all visible nodes are either matches or landmarks.

Browsing

A left mouse click on a node in the graph selects and displays the schematic represented by the node. A middle mouse click pops up a small window with auxiliary information about the schematic. (The auxiliary information could be any information relevant to the schematic such as page numbers in hardcopy editions, help, engineers' annotations, etc.)

The browsing environment has a Browser Window (see Figure Four) which displays user options and an alphabetical text list of the names of the components in the schematic currently displayed.

The schematics themselves are active. The user can click on a component to accomplish a variety of tasks such as further browsing (described in the following section), inspecting the underlying object in the knowledge base, filtering out the display components of its type from the schematic, showing a history of attached sensors, etc.

The user may also reformulate a query, refining

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

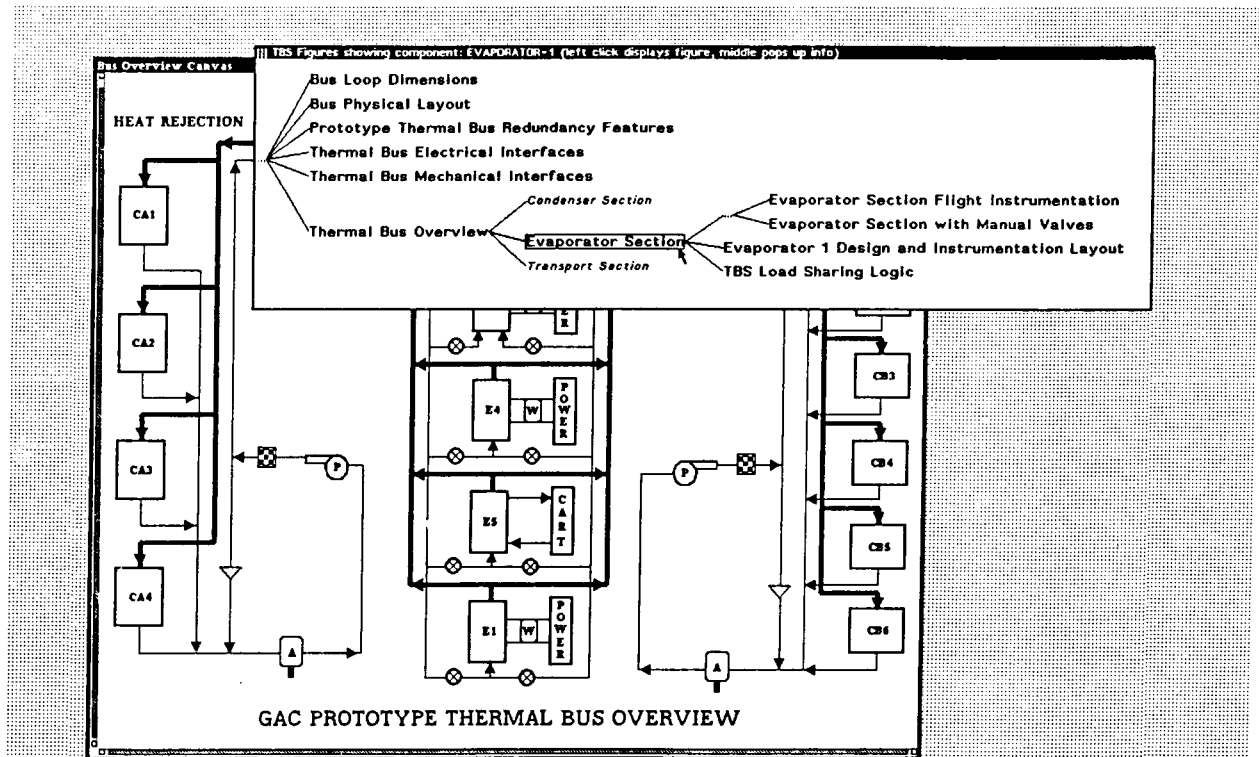


Figure Three. Filtered graph showing matches, in bold, for query requesting schematics with Evaporator-1. Landmarks, in italics, are a constant feature of the graph, giving a consistent shape to the graph for better user orientation.

it by asking for schematics with different or additional components. The user refers to components via a mouse selection on the picture of the component in the schematic, or by clicking on the name of the component in the text list. For example, the user may ask for only those schematics showing Evaporator-1, and refine the query to ask for schematics showing both Evaporator-1 and Evaporator-4. Many extensions for other kinds of queries can be envisioned, such as adding a perspective to the query, e.g., only those schematics of mechanical views showing Evaporator-1.

Adding Further Context

The query-filtered graph normally shows the user matches, landmarks and ancestors. However, the user may desire additional context to support the browsing activity. By its very nature browsing is interactive, and a user or intelligent system may not find the desired information the first time. At this point it may be helpful to show nearby nodes which are closely related to the actual matches. The Schematic Browser has an option for including *siblings* and *children* or just siblings of matches in the graph, as in Figure Five. The inclusion of sibling and child nodes develops a sense of *local context* for the user, so that nodes closely related to those that match the query are available for further exploration and to aid in the interpretation of the matches.

We also show a sort of dynamic local context by including the *previous matches* in reformulated queries, that is, the set of matches from the query immediately previous to the current one (Figure Six). For example, when the user reformulates the query for "Evaporator-1" as "Evaporator-1 and Evaporator-4", the previous matches from the query "Evaporator-1" are included. Thus the shape of the graph does not

change very much, helping to maintain consistency and orientation.

Other schemes for showing local context are of course possible, and may depend on application-specific criteria.

Presentation Control

In looking at the Figures in this paper the reader has certainly noticed the use of differing fonts to express node type on the graph. The fonts express *emphasis* and *de-emphasis* of the nodes, depending on their current status in the graph. Matches are emphasized - shown in large bold. The landmarks are smaller, italicized. The local context nodes are smaller still. The intention is to sharpen focus for the user by using: large fonts to draw the eye toward the nodes most likely to be of interest; smaller fonts to help reduce visual clutter; and the small italicized font to provide low-key emphasis for landmarks.

The user also has the ability to control the actual title used for the nodes. Optionally, shorter forms of the titles can be used for the de-emphasized nodes. In addition, page numbers (for the hardcopy versions of the schematics) are optionally shown.

To preserve the shape of the graph as much as possible, we maintain some standard order in the layout of the graph by sorting siblings alphabetically. This gives a feel of stability to the viewer so that nodes appear in the same context as new queries are formulated. (The alphabetical ordering is not optimal for the application and we will move to some new standard ordering to capture something of the semantics of the application, e.g., "Evaporator Section", "Transport Section" and "Condenser Section" in that order to reflect the functional flow of liquid in the bus.)

Again, many variations on the presentation

ORIGINAL PAGE IS
OF POOR QUALITY

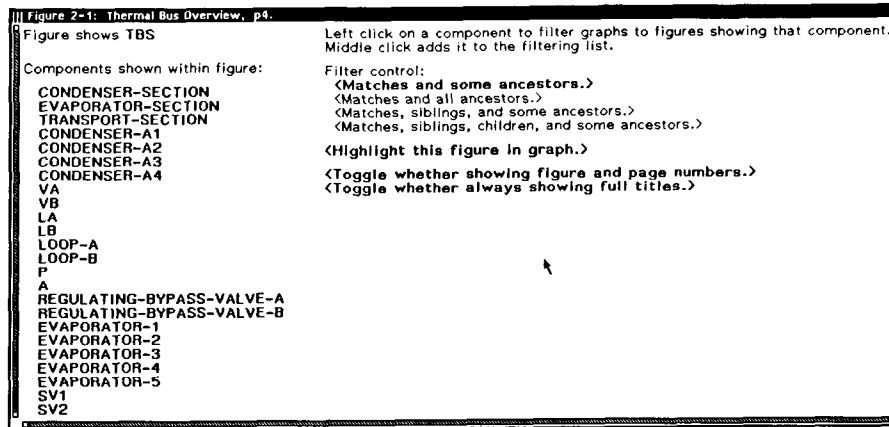


Figure Four. Browser Window for selection "Thermal Bus Overview". User options and a list of components in the schematic are displayed. The user may click on a component in the list for further querying.

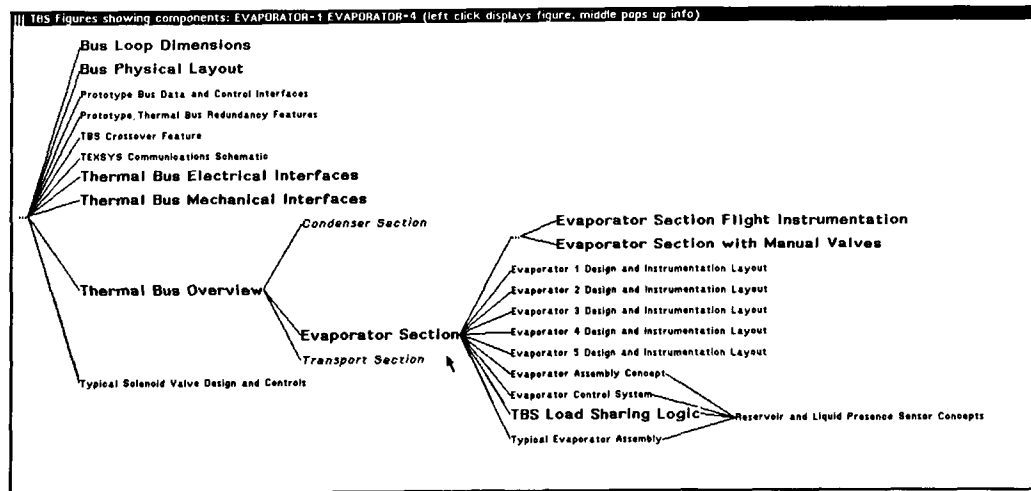


Figure Five. Filtered graph showing matches for query "Evaporator-1 and Evaporator-4", and siblings of matches, e.g., sibling node "Evaporator 1 Design and Instrumentation Layout".

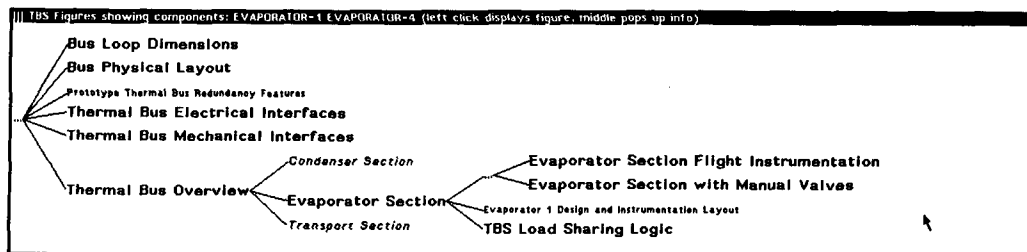


Figure Six. Filtered graph showing matches for query "Evaporator-1 and Evaporator-4", without siblings. Previous matches, nodes "Evaporator 1 Design and Instrumentation Layout" and "Prototype Thermal Bus Redundancy Features", de-emphasized in small font, are displayed from previous query for "Evaporator-1" (Figure Three).

control theme are possible. Experiments with the use of color or highlighting would prove interesting.

Fisheye Views

Our work with filters and context management in graphical browsers falls into the category of generalized fisheye views described by Furnas [10]. The basic notion of a fisheye view involves a balance of local detail and global context. Furnas identifies the components of a fisheye view as a focus node and a degree of interest function composed of an *a priori* interest function and a measure of distance from the focus node. The degree of interest function is computed for the nodes of the graph, given the focus node, and the top *n* nodes are shown.

Our work fits this model, with some differences. First, we utilize a *set* of focus nodes -- the matching subset determined by the query. We do not have a notion of showing just the top-*n*-ranked nodes. We emphasize *a priori* interest rather than distance in selecting nodes for display, that is, landmarks, ancestors and previous matches. Only the optional inclusion of siblings and children one node away from matches is based on a distance measure. Our *a priori* interest function has a dynamic aspect not found in Furnas' work, in preserving the previous matches in the graph so that the current view is in part a function of its history. Our graphs tend more toward stability and consistent shape both in the long term because of the use of landmarks, and in the short term via the inclusion of previous matches. Feiner [8] notes that the drastic changes in the shape of a fisheye graph which commonly occur may be disorienting, so consistency aids offset a disadvantage of the fisheye approach.

Also, in our work, we have the addition of the use of presentation control for emphasizing and

de-emphasizing nodes. This seems to fit in well with the intuitive notion of a fisheye in which the eye is drawn to prominent parts of the view because of their size and clarity, while less important parts appear to be less distinct and more distant.

Integration of the Schematic Browser with an Intelligent System

The Browser attempts to manage information presentation in a way that is supportive of the human user performing a browsing task. But significant portions of the browsing task may be done by an intelligent system which is monitoring activity in the domain knowledge bases and coordinating displays for the user.

As an example scenario of how this might work, the intelligent system and an astronaut are trouble-shooting a problem that appears to involve degraded condenser capacity. The intelligent system recognizes that the astronaut is likely to need a schematic view of the relevant subsystem. The intelligent system:

- Formulates an initial query, such as "Schematics showing condensers, their temperature sensors and the temperature sensors for the vapor lines".
- Chooses the options for the query-filtered graph -- whether to show children and siblings, full titles, etc.
- Chooses a schematic for initial viewing from the set of schematics which match the query.
- Annotates the schematic with messages and related displays, such as a temperature-time graph for the vapor line sensor whose reading is abnormally high.

The user can then proceed to use the selected schematic or refine the query and continue

**ORIGINAL PAGE IS
OF POOR QUALITY**

browsing as needed.

SUMMARY

The complexity of today's engineered systems makes heavy demands on engineers and others who must access and interact with the massive quantities of information which describe the systems. Our paper has discussed some techniques for accessing the kind of schematic drawings used routinely in engineering problem solving.

Problems of information access include having too many choices from which to select something interesting and relevant, and losing track of where a choice fits into the overall organization of the information. In our browser, choices are presented in a graph to provide the user the orienting context inherent in the relations between the nodes. An important part of the work of the Browser is to present only those nodes likely to be useful to the user.

Our general approach is to filter the nodes down to a small relevant subset, and to then add back in selected nodes for context. The nodes in the graph are filtered by user-formulated queries, reducing the choices to a more manageable and relevant subset. The Browser establishes context for the nodes matching the query by adding landmarks, ancestor nodes, previous matches and optionally, siblings and children of the matches. The Browser's presentation control mechanisms help the user to focus on the most important nodes through emphasis of the matching nodes and de-emphasis of context nodes.

ACKNOWLEDGMENTS

We would like to thank the following people for their valuable feedback: Jane Malin of the NASA Johnson Space Center; and Conrad Bock, Brian Drummond, Victoria Gilbert, Catherine

Perman and Mike Williams of IntelliCorp.
Errors are our own.

REFERENCES

1. Rasmussen, Jens, Information Processing and Human-Machine Interaction, Elsevier Publishers, New York, New York, 1987.
2. Davis, Randall, "Robustness and Transparency in Intelligent Systems", Proceedings Human Factors in Automated and Robotic Space Systems, National Research Council, Washington, D.C., 1987, 211-233.
3. Glazer, Keith, Southern California Edison, personal communication.
4. Kieras, David, "Guidelines for Diagrammatic Displays of Engineered Systems", unpublished manuscript, Ann Arbor, University of Michigan, 1987.
5. Bock, Conrad, "Visual Indexing and Interpretation or What's the Difference Between Frames, Logic, and Model Editors?", unpublished manuscript, Mountain View, CA, IntelliCorp, 1988.
6. IEEE Spectrum Staff, "Too Much, Too Soon: Information Overload", IEEE Spectrum, June 1987, 51-55.
7. Conklin, Jeff, "Hypertext: An Introduction and Survey", IEEE Computer 20:9, September 1987, 17-41.
8. Feiner, Steven, "Seeing the Forest for the Trees: Hierarchical Display of Hypertext Structure", Office Information Systems, Robert B. Allen, ed., ACM, 1988, 205-212.
9. Donelson, William, "Spatial Management of Information", Computer Graphics 12:3, August 1978, 203-209.
10. Furnas, George, "Generalized Fisheye

Views", Proceedings CIII '86 Human Factors in Computing Systems, Boston, April 13-17, 1986, 16-23.

11. Wise, John A., "Display Systems for Electrical System Control Centers", Proceedings of the Human Factors Society - 30th Annual Meeting, 1986, 1264-1268.
12. Shaw, John A., "Reducing Operator Error in Distributed Control Systems", InTech, March 1988.

ORIGINAL PAGE IS
OF POOR QUALITY

Presentation Planning Using an Integrated Knowledge Base*

Yigal Arens
Lawrence Miller
Norman Sondheimer

USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292

1 Introduction

ISI is involved in user interface research aimed at bringing together multiple input and output modes in a way that handles mixed mode input (commands, menus, forms, natural language), interacts with a diverse collection of underlying software utilities in a uniform way, and presents the results through a combination of output modes including natural language text, maps, charts and graphs.

Our system, **Integrated Interfaces**, derives much of its ability to interact uniformly with the user and the underlying services and to build its presentations, from the information present in a central knowledge base. This knowledge base integrates models of: the application domain (Navy ships in the Pacific region, in the current demonstration version); the structure of visual displays and their graphical features; the underlying services (data bases and expert systems); and interface functions. The emphasis in this paper is on a presentation planner that uses the knowledge base to produce multi-modal output.

There has been a flurry of recent work in user interface management systems (we list several recent examples in the references). Existing work is characterized by an attempt to relieve the software designer of the burden of handcrafting an interface for each application. The work has generally focused on intelligently handling input. In our paper we deal with the other end of the pipeline - presentations.

1.1 Presentation Planning

Presentations are put together by a **Presentation Planner**. The presentation planner decides what output mode or combinations of output modes to use for each piece of information. This involves recognition of the topic of the information, classification of the topic, a check of the user's

*This research is supported by the Defense Advanced Research Projects Agency under Contract No. N0014-87-K-0130. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

ORIGINAL PAGE IS
OF POOR QUALITY

preferences for presentation, and a coordinated delegation activity to assign tasks to the various output modes. This is done by rules that map between concepts and display modes.

In moving from an interface with a single output device to an integrated multiple output device interface, output processing changes substantially. Even in single-mode systems, we find that some preparation is necessary beyond the mere determination of the contents of presentations. For example, an information retrieval system may use tables exclusively for the display of retrieved data. Such a system may still decide to split information between tables in a report to control the length of the tables before the final output is generated. In an integrated presentation system, such planning activity grows considerably. The system must be able to decide what output mode to use for each piece of information.

The research issues that must be addressed in this context include determining what constitutes a good presentation of information, how to recognize information presentation situations, how to build knowledge that can be shared across several modalities, and how to choose the mode and form of output.

1.2 Planning as a Paradigm

In Presentation Planning, the use of the term *planning* is intended to bring to mind the AI sense of planning, where a system attempts to achieve a goal, executing certain operations at its disposal. The goal of our presentation planner is the presentation of some information, e.g., the status of a fleet of ships. The actions are commitments to present part of that information in some form, e.g., to present a map of the ships' positions showing the direction and speed of each by the direction of arrows, while showing their sailing schedules in associated text.

Certain constraints (called *grouping constraints*) must be considered in the process. In our naval demonstration domain, one constraint is that ships traveling together as a unit (*task force*) must be shown with a single symbol. Another is that ships in port are shown in a way that depends on how "familiar" the port is to the anticipated viewer. So while we never have the case where some ship in port is shown on a map while others in the same port are shown in a separate table, ships in different ports may well be shown differently.

We must also allow for coordination and cross reference. Text in a graph and in a natural language explanation should try to use similar vocabulary. The text might need to refer to the part of the map it is describing. This requires that the planning for individual media must be given enough advice by the overall planner to assure consistency and coordination.

We are exploring the planning paradigm by developing rules for good presentations and expressing them in the formalism of an AI planner.

2 Knowledge Bases & Rules For Presentation Planning

Presentation planning is achieved in our system by the application of a system of antecedent-consequent rules. The rules are used to classify the information that needs to be presented and to map types of information to appropriate types of presentations. Specifically, rule application involves *realizing* the categories that a given piece of information fits within, i.e., finding the rules whose antecedents describe the information; *selecting* the most germane category for the information, i.e., finding the most specific rules; and *redescribing* the information in appropriate textual and visual forms, i.e., using the consequents of the rules to structure the presentation.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

We cannot at this point claim that we have a complete theory of what constitutes a good presentation, since such a theory would have to explain aesthetic considerations involved in the preparation of presentations. While we cannot handle such considerations in general, we have been able to provide heuristics useful in certain situations. The Integrated Interfaces system contains rules that structure forms so that they contain what we consider appropriate amounts of information. Users whose aesthetic judgements differ from ours can modify these explicit rules to achieve different behavior. In this sense our system can be considered a presentation *shell*.

2.1 Example

The U.S. Navy's Pacific Fleet prepares a daily report on the situation and plans of the fleet. This report conveys current ship locations, courses, current activities, and the activities planned for ships in the near future. The person putting this situation report together has available for presentation a graphics system for ocean surface maps, a business graphics system for time tables, and methods for adding text to maps and tables.

Such a report could be presented in many ways. A map with lines showing each ship's course with a label at each point where the ship starts a new activity; or a map with points showing each ship's initial location and a timetable for each ship; or a map with points showing each ship's initial location and a label in English explaining its sailing plans. The Pacific Fleet uses the third form.

The Navy's report-generating activities can be described as following a process and rules similar to those encoded in our system. Information concerning ships is realized as belonging to certain known categories, e.g., the ship's planned activities. Rules for translating such information into a component of a report, e.g., an indication on a map or a textual description, are then examined, and a rule appropriate for the desired mode of presentation is selected. The information about the ships is then redescribed as part of the presentation being prepared.

2.2 Design

2.2.1 Models

Our models characterize or define the categories of *entities* our user interface can deal with. One of the models identifies the categories of objects and actions in a common-sense view of the domain of our system. We indicate subclass relations present between categories, as well as relationships between objects and actions. For the Navy application we described above, we include the various categories of ships and sailing activities. We also include specific knowledge, such as that *Tankers* are a type of *Ship*, and that a *Repair* activity involves a *Disabled Ship*.

Another model describes the categories of objects and actions of the interface world. The objects here include windows, tables, maps, text strings, and icons. The actions here include creation, deletion, movement, and structure of displays.

A final model (not crucial for this discussion) describes the functions and data structures of the available application services. Included here are descriptions of underlying application software, and any database schemas.

2.2.2 Rules

The presentation rules are simple: they map object from the application domain model into objects in the interface model. So the entity that describes a daily status report may be mapped into a

map. A position report may be mapped onto a point. A ship's planned future activities may be mapped onto a text string.

These rules are arranged according to the class subsumption hierarchy of the models. So the rules applicable to all ships are further up the hierarchy than those applying only to tankers.

A system that constructs a visual display based entirely on an analysis of the details of the data to be presented (cf. Mackinlay [5]) holds considerable appeal. However, in a domain as complex as ours, it is probably impossible to design such a presentation system. We thus allow both "low-level" rules, such as those that map the various types of ships to their icons, and "high-level" ones, which given a particular type of presentation request provide a script to be followed in fulfilling the request.

2.2.3 Rule Application

Presentation planning can now be described as the task of recognizing the domain categories within which a request for information presentation falls, selection of the appropriate rules that apply to those categories, and mapping of the domain terms in the request into appropriate presentation terms.

The three phases which we refer to as *realization*, *selection*, and *redescription* are implemented in our system as described below.

Realization relates the facts about instances to the abstract categories of the model. For example, the concrete facts about *Sprite*, a ship with a malfunctioning radar, must lead to the realization that it is a *Disabled Ship* (assuming *Disabled Ship* is defined in the domain model). Selection works by allowing for the appropriate mapping rules to be chosen, allowing for additivity. Selection also assures that all aspects of the demand for presentation are met by some rule. Redescription applies the rules, mapping each aspect of a common-sense view of a presentation into an equivalent presentation form.

The forms produced by rule application are not actually the commands to the output subsystems (i.e., the map graphics system, text generator, and the business forms system). Instead, they are interpretable by *device drivers* that control these systems. This design allows the forms produced by the rules to serve as a model for the contents of the screen. Although we do not currently do so, user input activity on the screen could be interpreted with this screen model serving as a context. So our design has the additional advantage of allowing, in principle, the use of the same knowledge base and many of the same inference mechanisms for analysis and presentation planning.

3 Knowledge Representation Tools

Our implementation of presentation planning depends on two knowledge representation systems: NIKL and KL-TWO. NIKL holds our models. KL-TWO automatically carries out realization. KL-TWO also holds the demands for presentation and receives the forms read by the device drivers. This section provides a brief introduction to these tools.

3.1 NIKL

NIKL [3] is a network knowledge-base system descended from KL-ONE [1]. This type of system supports description of the categories of entities that make up a domain. The central components

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

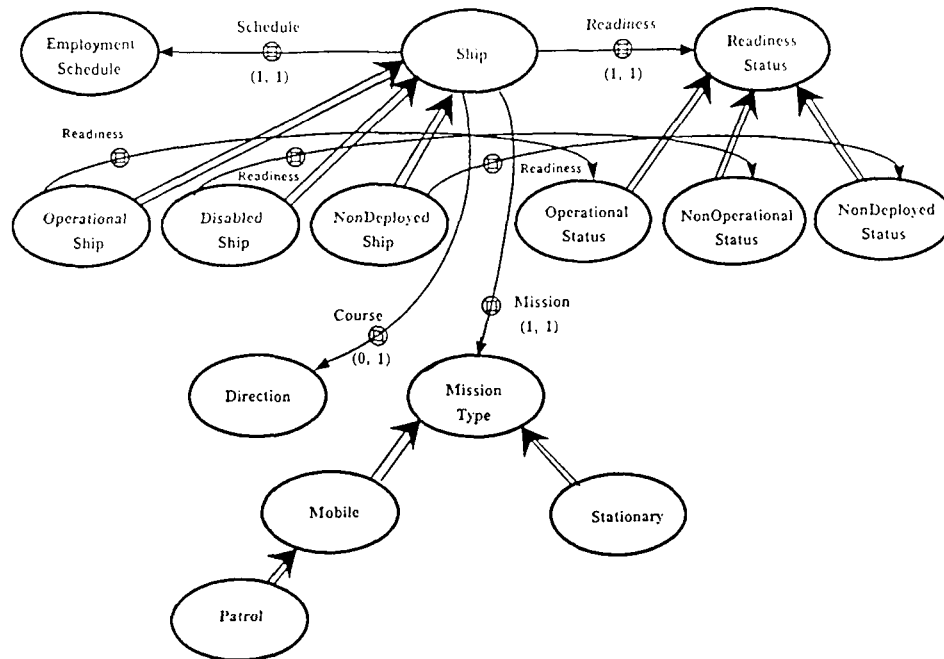


Figure 1: Fragment of Domain Model Containing *Ship*.

of the notation are sets of concepts and roles, organized in IS-A hierarchies. These hierarchies identify when membership in one category (or the holding of one relationship) entails membership in (or the holding of) another. The roles are associated with concepts (as *role restrictions*), and identify the relationships that can hold between actual individuals that belong to the categories. The role restrictions can also hold number restrictions on the number of entities that can fill these roles.

We have been experimenting with a naval assets domain model for the naval briefing application mentioned above. It has a concept *Disabled-Ship* that is meant to identify the ships that are unable to carry out their missions. *Disabled-Ship* IS-A type of *Ship* distinguished from *Ship* by having a role restriction *Readiness* that relates *Disabled-Ship* to *NonOperational-Status*, i.e., all ships with nonoperational status are disabled. All *Ships* can have exactly one filler of the *Readiness* role restriction. The concept of *NonOperational-Status* is partly defined through the IS-A relation to a concept *Readiness-Status*. This situation is shown graphically in Figure 1 in the typical network notation used for KL-ONE knowledge bases.

In flavor, NIKL is a frame system, with the concepts equivalent to frames and the role restrictions to slots. However, the NIKL representation can be given a formal semantics. In fact, we could translate our NIKL knowledge bases into predicate calculus expressions and use a theorem prover to make the same inferences we do. However, NIKL is optimized for the limited inferences it makes and a general purpose theorem prover would be less efficient.

3.2 KL-TWO

KL-TWO is a hybrid knowledge representation system that takes advantage of NIKL's formal semantics [8]. KL-TWO links another reasoner, PENNI, to NIKL. For our purposes, PENNI, which

is an enhanced version of RUP [6], can be viewed as restricted to reasoning using propositional logic. As such, PENNI is more restricted than those systems that use first order logic and a general purpose theorem prover.

PENNI can be viewed as managing a data base of propositions of the form $(P\ a)$ and $(Q\ a\ b)$ where the forms are variable free. The first item in each ordered pair is the name of a concept in an associated NIKL network and the first item in each ordered triple is the name of a role in that network. So the assertion of any form $(P\ a)$ is a statement that the individual a is a kind of thing described by the concept P . The assertion $(Q\ a\ b)$ states that the individuals a and b are related by the abstract relation described by Q .

NIKL adds to PENNI the ability to do taxonomic reasoning. Assume the NIKL database contains the concepts just described in discussing NIKL. Assume that we assert just the following three facts: $(Ship\ Sprite)$, $(Readiness\ Sprite\ C4)$ and $(NonOperational-Status\ C4)$; $C4$ is a U.S. Navy readiness code. Using the knowledge base, PENNI is able to deduce that any *Ship* whose *Readiness* is a *NonOperational-Status* is a *Disabled-Ship*. So if we ask if $(Disabled-Ship\ Sprite)$ is true, KL-TWO will reply positively.

PENNI also provides a truth maintenance system that keeps track of the facts used to deduce others. When our rules are used to determine aspects of a presentation from facts about the world, the truth maintenance system records the dependencies between the domain and the presentation. For example, $(Readiness\ Sprite\ C4)$ triggers a rule which asserts $(Disabled-Ship\ Sprite)$. If $(Readiness\ Sprite\ C4)$ is retracted, PENNI's truth maintenance system will automatically retract the assertion that the *Sprite* is a disabled ship.

4 Examples

The power of Presentation Planning is in its flexibility. The designer of a system does not specify rigidly in advance in what form information will be requested from the user, and how data and results will be displayed. Instead, our models contain descriptions of the types of information the application programs deal with, and of the types of graphical tools and instruments available. The rules for presentation enable the system to generate on-demand displays appropriate for given needs. Here are some concrete examples.

4.1 Construction of a Visual Representation of an Object

Consider the knowledge about ships and about graphical instruments encoded in the NIKL models in Figures 1 and 2. Besides the aspects of Figure 1 already indicated, note that *Ships* have *Missions* and that *Patrol* missions are a subclass of *Mobile* missions. Note also that all *Ships* have *Schedules*. Figure 2 describes some *Graphical-Instruments*. This includes *Text* for language output, *Icons* for maps and isolated forms, and *Visual-Enhancements* that could apply to icons and text. *Icons* have *Text* as their *Tag*. Several specific *Icons* and *Visual-Enhancements* are included.

Let us assume that the user wishes to show ships engaged in a *Mobile* mission with a special *Icon*, and that the icon should be oriented in a direction identical to the ship's course. In addition, assume that *Disabled-Ships* are to be shown with *Red* icons and that the *Schedule* of a ship is to be shown in the natural language *Tag* of the *Icon* representing it. A version of the rules that we would use to achieve this is shown in Figure 3.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

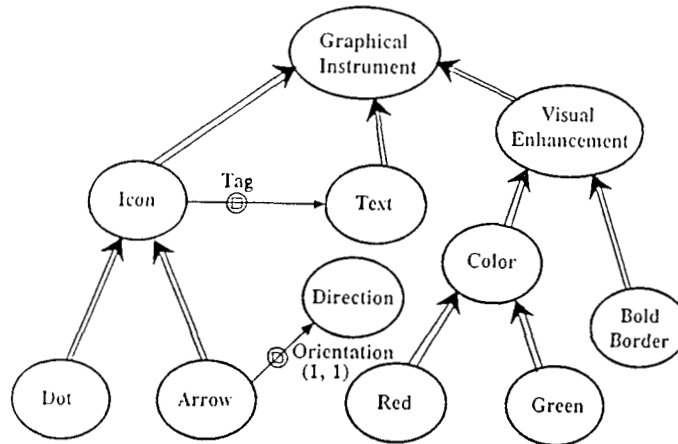


Figure 2: Fragment of Interface Model Containing *Graphical-Instrument*.

-
1. IF (Operational-Ship x) or (NonDeployed-Ship x)
THEN (Icon-Color Image(x) Green)
 2. IF (Disabled-Ship x)
THEN (Icon-Color Image(x) Red)
 3. IF (Ship x) and (Mission x y) and (Course y z)
THEN (Orientation Image(x) z)
 4. IF (Ship x) and (Mission x y) and
(Mobile y) THEN (Icon-Type Image(x) Arrow)
 5. IF (Ship x) and (Schedule x y)
THEN (Tag Image(x) Textual-Description(y))
-

Figure 3: Sample Presentation Rules

The antecedent considers the categories of one or more individuals and their relationships, all in terms of the NIKL models. The consequents provide assertions about the graphic representation of objects for the PENNI database. These rules are asserted into PENNI so that the truth maintenance system may keep track of the dependencies between antecedent facts and their resultant consequents, as explained in the previous section.

The functions *Image* and *Textual-Description* map the constants of the common sense world into constants of the visual and textual world, respectively. For example, Rule 5 above states that if some individual, *x*, is a *Ship* and another individual, *y*, is its *Schedule*, then the *Tag* of the image of *x* is the textual-description of *y*. The textual-description of *y* will be created by the invocation of our text generator.

To complete the example, suppose that the following set of facts was asserted into the PENNI database: (*Ship Sprite*), (*Readiness Sprite C4*), (*NonOperational-Status C4*), (*Mission Sprite X97*), (*Patrol X97*), (*Schedule Sprite U46*), (*Course X97 220*), and (*Employment-Schedule U46*). Suppose further that the NIKL model defined *Patrol* to be a subclass of *Mobile* missions. Then realization would recognize the 'Sprite' as a *Disabled Ship* and one engaged in a *Mobile* mission on a course of 220 degrees. Selection would identify that Rules 2, 3, 4 and 5 apply. Redescription would result in the addition to the PENNI database of the description of the image of the 'Sprite' as a red arrow with an orientation of 220, and with a textual representation of its schedule as its label.

If any of the facts pertaining to *Sprite* is retracted, an automatic change in the description of its graphic image will occur.

4.2 Recognizing Special Cases

For many requests for information encountered in our domain the presentation required is far more complex than the rules of the kind listed above could provide for. The construction of these complex presentations requires, among other things, a global evaluation of the coherence of the display. It would therefore be hopeless, at this point, to attempt to write rules that would attempt to derive an elaborate presentation entirely from low-level information about the objects to be described. Our approach provides us with a partial solution to this problem.

The availability of models of the domain and of displays to our Presentation Planner gives it the advantage of being able to recognize collections of data as representing information of a certain known type. The Presentation Planner can then make use of presentation techniques specialized for this type of data to provide the user with more appropriate displays.

For example, Figure 4 provides portions of our model that include the class *Pacific Situation*, a display of data about ships and ports in the *Pacific Region*, which includes certain specific information from the ships' employment schedule.

When provided with data about ships in the Pacific region and their employments, the Presentation Planner would classify the data in its model of the domain. A spatial reasoner deduces the region containing all of the ships which would be included in the *Pacific Region* and the Presentation Planner recognizing that it has received a collection of data belonging to the class *Pacific Situation*. Once the classification of the data is accomplished the Presentation Planner will use specific presentation rules appropriate for displaying the information. In the domain we have considered there is a preferred way for presenting this information, to which we try to conform. This preferred presentation has developed in the Navy in the course of years of handcrafted situation briefing presentations.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

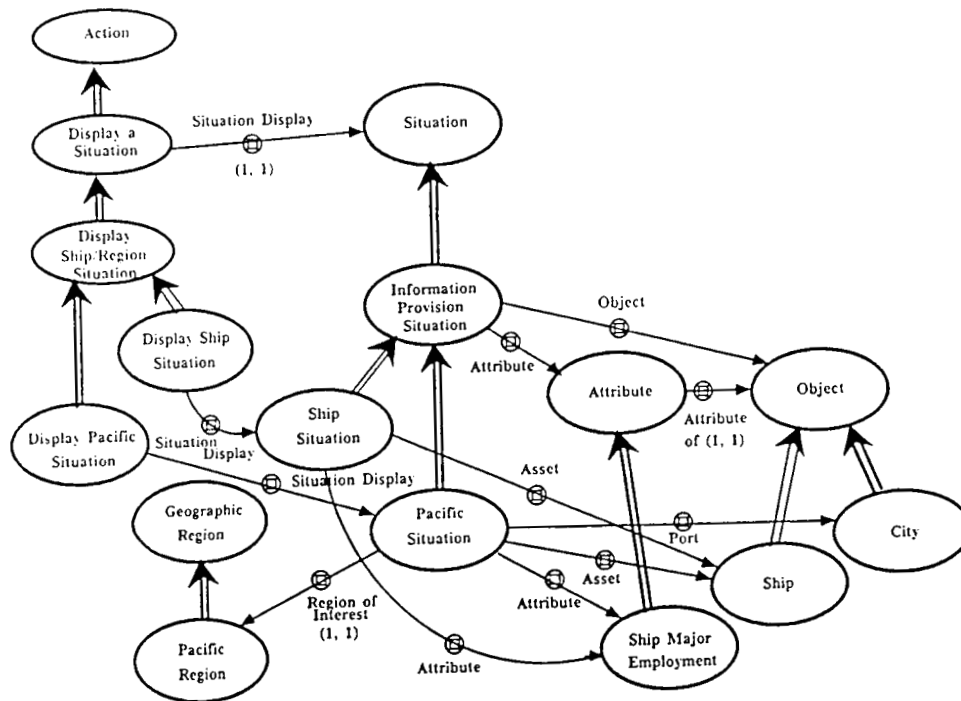


Figure 4: Fragment of Domain Model Including *Situation*.

The specific presentation rules appropriate only for a situation briefing will combine the entities created by more general rules, of the kind described in the previous section, to produce the final presentation.

4.3 Generation of an Input Display

The Presentation Planner must also deal with the preparation of displays for the purpose of soliciting necessary information from the user. Here, again, the models of all aspects of the task and the domain are indispensable.

At some point the user may indicate a desire to view data concerning ships in some region. In terms of our model (see Figure 4), that would mean indicating a preference for *Display a Situation*. As it turns out, the Presentation Planner does not have any rules that can be used to redescribe this general request into a presentation, but there exist ways of satisfying more specific requests. For example, requests to have the *Pacific Region* or any of its subregions displayed can be satisfied. As we see in Figure 4, the situation involves specific ships and ports, which may also be displayed.

In this case, the Presentation Planner collects all options the user can choose among to construct an executable request. The Presentation Planner then plans a display form that will be used to present these options to the user. The result of this plan is a set of assertions in PENNI that the device driver for a separate form management package (QFORMS) [2] will use to prepare the input form.

The form below, presented to the user, allows the user to make one of several specific choices:

Pacific Regions:

Western Pacific	<input type="checkbox"/>
South China Sea	<input type="checkbox"/>
Indian Ocean	<input type="checkbox"/>
Eastern Pacific	<input type="checkbox"/>
Pacific Command Region	<input type="checkbox"/>

Ship: _____

It is instructive to examine precisely how this form is created. Specifically, how does the choice "Ship" become part of the form? It is not a Pacific Region, but Navy personnel request that this possibility be supported.

We thus included in our model the concept *Display Ship/Region Situation*. Since this has two subclasses of actions, namely *Display Ship Situation* and *Display Regional Situation* our system considers the possibility of generating an intermediate two item submenu, something like:

Situation in Pacific Region	<input type="checkbox"/>
Situation of Ship	<input type="checkbox"/>

We considered this unsatisfactory from a human factors standpoint. We therefore formulated a general rule saying that if the choices on a proposed menu can be further subdivided, and if the number of choices is less than N , then the proposed menu should not be displayed. Instead, a more detailed form should be generated, one based on the subchoices. Our prototype uses the value 3 for N , so in this case the rule causes the Presentation Planner to immediately generate the more specific form. A user is free to change the value of N , thus modifying the design of forms the system generates in situations like the one above.

Note that the geographic regions available were specified by name in the form created, while ships were not. Rather, the user is allowed to specify the desired ship by typing it on the form (Figure 5). This distinction is a result of information concerning the cardinality of the relevant collections of objects – information encoded in our models. Since the number of possible choices for region is small, they are enumerated. However, the number of ships is larger, so the user is provided with a way to specify a choice explicitly instead.

Generating interfaces by models and rules is time consuming and tedious. But it forces the designers to think out every aspect of an interface. The decisions are not hidden in the code, they are explicit – observable, modifiable – in the rules and the model.

5 Related Work

The literature contains numerous examples of User Interface Management Systems. However, we see our contribution as being our emphasis on *Presentation Planning*, and very few systems are concerned with this aspect of the interface. Perhaps the best known previous work dealing with this issue is that of Mackinlay [5].

Much like part of our system, Mackinlay's *APT* uses information about characteristics of data provided to it, to produce a graphical representation of that data. The differences between the two systems become clear when we consider the variety of data each deals with and the variety of presentations they produce. *APT* produces graphs of various kinds, and much of its effort goes into deciding which axes to choose, and how to indicate the values along each axis. Data dealt with is limited to what can be presented using such graphs. Consequently, Mackinlay has succeeded in producing a system which can generate graphical presentations automatically using only "low-level" information about the objects and their attributes.

Our system is expected to generate a much wider variety of displays, many that would require considerable design work even from an expert human graphic artist.¹ In addition, certain display layouts are often chosen simply to conform to pre-existing preferences of Navy personnel. Consequently, unlike Mackinlay, we are required to provide for the possibility of following pre-set stereotypical instructions in certain cases. We thus must devote considerable effort to recognizing which cases require these special displays.

A further significant difference between the systems is the complexity of the data we are required to present. In order to handle this range of data we must represent it using a sophisticated knowledge representation language, NIKL, a facility which Mackinlay finds unnecessary in *APT*. Both systems make use of sophisticated reasoning facilities.

6 Problems

We believe our approach to the problem of presentation planning is a viable one. Indeed, as illustrated in the examples of the previous section, we are using it to generate various interesting displays. However, there are still numerous outstanding problems which remain to be solved. In this section we will list some of the more difficult and interesting ones; some of them are inherent to presentation planning while others are specific to choices we have made in our system.

¹As in fact they do. Maps of the kind produced by our system take Navy personnel approximately 4 hours to produce every day.

6.1 Lack of Coordination between Output Modes

Currently, we are using off-the-shelf programs for the low level production of output. This places us in a position of having to divide our data between the available facilities without having access to the internal decisions made by those facilities. In reality linguistic considerations may play an important part in the decision to use a pointing gesture. For example, in a situation where using language to describe an object to the hearer would be difficult or awkward, we prefer to point to it. Our existing setup does not permit the Presentation Planner to become aware of such difficulties.

The proper solution to this problem would probably require a uniform approach to all methods of communication and a more complete understanding of their relative capabilities. This appears to be a hard problem. We are not aware of any existing efforts in this direction.

Related to the problem mentioned above is the question, Which information about presentation planning can be shared across media and modalities and which is unique to each medium?

6.2 Modeling Difficulties

The domain of graphical displays is not yet well understood. We are facing difficulties in developing a model that expresses all the information needed to plan presentations. Certain idiosyncracies of NIKL have added to the difficulty of representing some of the knowledge. Several of these problems will be resolved with the development of Loom [4] to which we intend to switch as soon as it becomes available.

6.3 Lack of a User Model

A user model will enhance the Presentation Planner. For example, knowledge about a user's familiarity with a certain geographic area will allow the system to label only unfamiliar ports and regions, thus reducing screen clutter. While incorporating a user model is in our longer range plans, we have not yet begun to do so.

6.4 Lack of a Dialogue Model

A dialogue model will allow the presentations to be more closely tailored to the user's requests. Currently, the Presentation Planner is simply provided with data to display. It is not aware of the purpose of the display, nor even of the user request that prompted it. Keeping track of such information is also in our future plans.

6.5 Complexity of Correspondence between Domain and Interface Models

Many of our presentation rules assume a simple correspondence between domain objects and their graphical icons. This may turn out to be an oversimplification. It might be necessary for us to posit intermediate levels between the domain and the display; a common-sense reasoning level, for example.

7 Current Status

A demonstration version of the Integrated Interfaces system is now available at ISI. The current version models the domain of Navy ships in the Pacific Ocean. A user may use the system to access

**ORIGINAL PAGE IS
OF POOR QUALITY**

ORIGINAL PAGE IS OF POOR QUALITY

information about ships' locations, tasks, readiness status, and more. The resulting information is displayed using combinations of maps, menus, tables, and natural language output (Figure 5).

The system is written in Common Lisp and runs in the X windows environment under UNIX on HP 9000 Model 350 workstations. Displays are presented on a Renaissance color graphics monitor. The map graphic modality is supported by ISI's Graphics Display Agent. Menus and forms are created using QFORMS [2]. Natural language output is produced by ISI's Penman system [7].

8 Acknowledgements

We wish to acknowledge the crucial help provided by others working on the Integrated Interface project. Stu Shapiro helped in developing the general framework of the system and contributes to its implementation. Paul Raveling has developed the graphical interface and continues to maintain the GDA. Chin Chee has ported QFORMS and Penman to the HP workstation and is responsible for coordinating the various parts of the system. Jim Geller has contributed to the implementation of the spatial reasoner.

9 References

- [1] Brachman, Ronald J. and James G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science* 9(2), 1985, pp. 171-216.
- [2] Kaczmarek, Tom, "CUE Forms Description," ISI Internal Report, USC/ISI, Marina del Rey, CA., 1984.
- [3] Kaczmarek, Tom, Ray Bates and Gabriel Robins, "Recent Developments in NIKL," *Proceedings, AAAI '86*, Philadelphia, PA., August, 1986.
- [4] MacGregor, Robert and Ray Bates, "The Loom Knowledge Representation Language," *Proceedings of the Knowledge-Based Systems Workshop*, St. Louis, MO., April, 1987. (Also available as ISI Reprint Series ISI/RS-87-188.)
- [5] Mackinlay, Jock D. "Automatic Design of Graphical Presentations," Ph.D. Thesis, Department of Computer Science, Stanford University, Stanford, CA., December, 1986.
- [6] McAllester, D. A., "Reasoning Utility Package User's Manual," Massachusetts Institute of Technology, AI Memo 667, Cambridge, MA., April, 1982.
- [7] Sondheimer, Norman K. and Bernhard Nebel, "A Logical-Form and Knowledge-Base Design For Natural Language Generation," *Proceedings, AAAI '86*, Philadelphia, PA., August, 1986, pp. 612-618.
- [8] Vilain, M., "The Restricted Language Architecture of a Hybrid Representation System," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA., August, 1985, pp. 547-551.
- [9] Walker, E., R. Weischedel and N. Sondheimer, "Natural Language Interface Technology," *Proceedings of the Strategic Systems Symposium*, Monterey, CA., October, 1985.

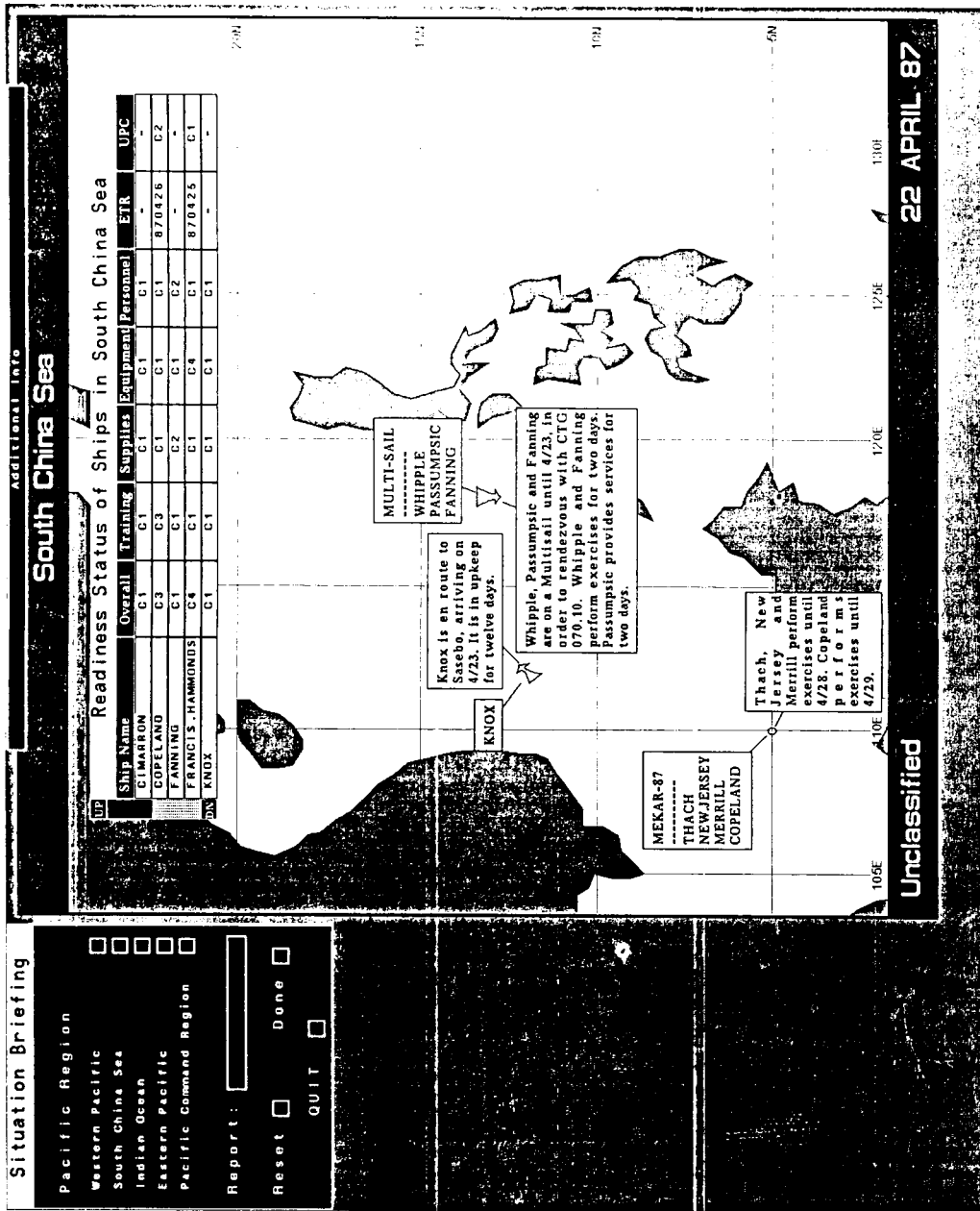


Figure 5: South China Sea Situation Display.

ADAPTATION IN HUMAN-COMPUTER INTERFACES

Sherman W. Tyler
Lockheed Missiles and Space Company

(Paper not provided by publication date.)

PHYSIOLOGICAL ASSESSMENT OF TASK UNDERLOAD

J. Raymond Comstock, Jr.
Randall L. Harris, Sr.
Alan T. Pope

NASA Langley Research Center
Hampton, Virginia 23665-5225

Abstract

The ultimate goal of research efforts directed at underload, boredom, or complacency in high-technology work environments is to detect conditions or states of the operator that can be demonstrated to lead to performance degradation, and then to intervene in the environment to restore acceptable system performance. Physiological measures may provide indices of changes in condition or state of the operator that may be of value in high-technology work environments. The focus of the present study was on the use of physiological measures in the assessment of operator condition or state in a task underload scenario. A fault acknowledgment task characterized by simple repetitive responses with minimal novelty, complexity, and uncertainty was employed to place subjects in a task underload situation. Physiological measures (ECG, EEG, and pupil diameter) were monitored during task performance over a one-hour test session for 12 subjects. Each of the physiological measures exhibited changes over the test session indicative of decrements in subject arousal level. While high correlations between physiological measures were found across subjects, individual differences between subjects supports the use of profiling techniques to establish baselines unique to each subject.

Introduction

With the development and increasing number of high-technology work environments, the roles, functions, and tasks performed by people in these environments need to be considered carefully. Although not unique to high technology or automated environments, words such as boredom, underload, and complacency describe conditions of the human operator or system manager that may have major impact on the performance and safety of high-technology systems.

Increased automation can change the nature of the task that the operator is performing, and may increase the monitor-

ing or vigilance component of a task (Parasuraman, 1987). Many studies and reports concerning the effects of aircraft automation may be found in the literature (e.g. Bergeron & Hinton, 1985; Wiener, 1985; Wiener & Curry, 1980). Each report echoes concern, both about errors made in the use of automated systems and failure of the operator or crew to detect errors when using an automated system.

Reports concerning the effects of automation on commercial aircraft pilots note that too much automation may result in complacency and boredom (Chambers & Nagel, 1985), and pilot reports of just "being along for the ride." (Chambers & Nagel, 1985; Wiener, 1985). Chambers and Nagel (1985) discuss potential alternatives in the allocation of functions between automated systems and the human operator. In one system, which they called the "pilot's assistant concept," the pilot plays an active role as the system manager. It was suggested that the active role of the pilot may serve to reduce complacency and boredom. A second system, the "electronic copilot concept," was presented in which the pilot functions as a passive monitor, with concomitant risks of complacency and boredom. In reviewing the report by Chambers and Nagel, Parasuraman (1987) suggested "that the goal of human factors engineering must not necessarily be to reduce workload but to optimize it."

In research on vigilance, conditions described as boredom, monotony, and fatigue have been reported as having overwhelming effects compared to the many variables (e.g., temperature, noise, circadian effects, illumination, and vibration) that have received experimental attention (Mackie, 1987). Studies examining stress effects on sonar operators show that boredom, monotony, and fatigue were ranked the highest of all the stressors considered (Mackie, Wylie, & Smith, 1985).

The ultimate goal of research efforts directed at underload, boredom, or complacency is to detect conditions or states of the human operator or crew member that can be shown to lead to performance degrada-

tion, and then to intervene in the automated work environment to restore acceptable system performance. While monitoring of machines by human operators has a long history, monitoring of the human operator by the machine, or better yet, an interaction or "dialog" between the human and machine is only becoming feasible through the technology of today and the promises and potential of the technology of tomorrow.

In order to detect changes in the condition or state of the human operator or crew member, physiological measures may provide the indices required. Many studies have been conducted exploring the relationship between physiological parameters and "workload," "vigilance," or "fatigue" (See O'Donnell, 1979, for a review), and physiologically driven "alertness indicators" (O'Donnell, 1979, pp. 57 & 62) have been suggested and explored. However, in order for these measures to be employed as viable indices of operator condition or state, research efforts must take advantage of new and steadily improving sensor technology, and state-of-the-art analysis methodologies. Horst (1988) discusses some of the areas of application and problem areas associated with physiological measures, and notes that on-line or real-time applications of physiological indices may include: (1) assessment of "the general state of the operator," in order to determine whether the operator should be "in the loop" at all, (2) dynamic task allocation between human and machine, and (3) detection of important events that the operator did not attend to or detection of events with error responses (Horst, 1988, p. 30).

The focus of the present study was on the use of physiological measures in the assessment of operator condition or state in a task underload scenario. Research directed at detecting conditions or states of the operator in a task underload scenario that can be related to performance degradation can be conceptualized as having four stages:

(1) Stage 1 involves establishing the sensitivity of selected physiological measures to the condition or state of the operator in task underload scenarios. This stage may be accomplished, as in the present study, through physiological measurement of subjects in laboratory or simulation environments. One question to be answered at this stage is whether there are generic measures that may be used as indicators for all human operators or whether individual differences between people require the development of profiles unique to individuals or groups of individuals.

(2) Stage 2 is concerned with establishing the relationship between observed physiological changes and task performance. Obviously, there may be many attributes to tasks in the high technology environment, involving a variety of types of cognitive processes on the part of the operator. For example, if physiological measures are indicative of operator "boredom," on which types of tasks can one look for degraded performance (e.g. perceptual, memory, problem solving / decision making). This stage may be investigated most easily by studies conducted in settings in which a high degree of control over the task is possible, such as in laboratory or simulation environments.

(3) Stage 3 involves establishing the relationship between physiological indices and performance in real-world settings. In order to accomplish this, performance assessment and physiological measurement may be conducted in instrumented aircraft or instrumented ground transportation vehicles. In order to investigate underload or "boredom" in real-world settings, it may be necessary to monitor operators or crew members over extended periods of time.

(4) Stage 4 involves the development and testing of off-line and on-line interventions. Such interventions or remediation strategies would be designed to insure that conditions or states of the operator associated with performance degradation are handled in such a way as to restore acceptable system performance.

The present study was directed at Stage 1, described above, and was focused on establishing the sensitivity and inter-relationship of heart rate and heart rate variability, pupil diameter, and electroencephalographic data during performance of a task in which the subject was underloaded. The characteristics of such a task, simple repetitive responses with minimal novelty, complexity, and uncertainty (Cooper, 1968), were such that boredom would be expected.

Methodology and Design

Subjects. Physiological and task performance data were obtained from 12 subjects (10 male; 2 female). Subject ages ranged between 20 and 44 years, with a median age of 28.5 years. Subjects were volunteers from among the staff and contract personnel of the NASA Langley Research Center.

Task. A fault acknowledgment task (five alternative choice-reaction-time task) was selected as a laboratory benchmark task for inducing boredom. The fault acknowledgment task consisted of a desk-

top-computer-generated jet engine pictorial in which one of five areas could be highlighted in red. The task for the subject was to detect the fault, then to press the appropriate response key for the highlighted area, which turned off the highlighted area. The highlighted area remained lighted until the subject responded. If the subject did not respond, a time-out occurred after about 30 seconds, and a new fault was presented. The task was presented on a CGA computer monitor and responses were made on the desktop computer keyboard.

Subject response times and accuracy of responding were recorded for each fault presentation. Frequency of presentation of faults, or intertrial-interval (ITI), for half the subjects was 6 seconds (randomly ranging between 4.8 and 7.2 seconds) throughout the test session and for the remainder of the subjects alternated between 20 second ITIs (range: 16 to 24 seconds) and 2 second ITIs (range: 1.6 to 2.4 seconds) with each ITI maintained for blocks of 5 minute duration. The alternating ITI condition was incorporated in the study in order to examine contrasts in the physiological data during periods in which the task made differing demands on the subject. A complete test session lasted approximately one hour and consisted of 12 five minute blocks. A time synchronization signal was sent from the desktop computer to a Digital Equipment Corporation (DEC) Modular INstrument Computer (MINC) system on which the physiological data were recorded.

Procedure. Upon arrival at the laboratory, subjects initially completed voluntary consent forms, then electrode attachment began. After electrode installation, subjects were seated in front of the task display where instructions on performing the task were given and calibration of the non-contact oculometer system used to measure pupil diameter was completed. The experimental session was then started. Time keeping devices (e.g. clocks and wrist watches) were removed from the experimental setting during the test session to minimize the subject's awareness of the time remaining in the test session. At the conclusion of the test session information was obtained from each subject concerning food, beverages, medications, and sleep prior to the test session. Subjects were also queried about task strategies or mental "games" that they engaged in during the session.

Subject self-report measures. Each subject completed the Jenkins Activity Survey (JAS) (The Psychological Corporation) and the Eysenck Personality Inventory (EPI) (Educational and Industrial Testing Service). The JAS provided an index of the Type A behavior pattern.

Prior studies have explored the relationship between the Type A behavior pattern and physiological responses, and, in particular, heart rate and blood pressure (Perkins, 1984; Lake, Suarez, Schneiderman, & Tocci, 1985). The EPI provided an index of introversion-extraversion.

Physiological measures

Heart Rate (HR) and Heart Rate Variability (HRV). The electrocardiogram (ECG) signal, from which HR and HRV were derived, was obtained through active electrodes attached to the top of the sternum and to the lower left rib cage. A reference electrode was attached to the left ankle. The ECG signal was fed through an opto-isolated bioamplifier to a schmitt trigger which produced a digital output upon an ECG signal positive-going threshold crossing (R-wave). The time interval between successive threshold crossings, or inter-beat interval (IBI), was recorded on the DEC MINC system.

Pupil Diameter (PD). The PD of each subject was measured using a non-contact infrared-light bright-pupil oculometer system. The oculometer electro-optical head was located 60-80 cm from the subject and to the left of the CGA color monitor presenting the task. PD was sampled at one-second intervals throughout the test session and recorded on the DEC MINC system.

Electroencephalography (EEG). EEG recording sites were over the occipital-parietal cortex (locations O1 and P3) using a bipolar lead configuration, with the reference electrode at the ipsilateral mastoid bone. The power (integrated amplitude) of the EEG signal in the various frequency ranges was obtained by filtering the signal and integrating the filtered output. The EEG signal was filtered into five frequency range components: (1) delta (1-4 Hz), (2) theta (4-8 Hz), (3) alpha (8-13 Hz), (4) beta1 (13-20 Hz), and (5) beta2 (20-40 Hz). The output of each filter was fed to a contour following integrator with a 1 second time constant (Coulbourn Instruments filters and integrators). The integrated amplitude of each of the five EEG frequency ranges was sampled by the DEC MINC system at one-second intervals throughout the test session.

Results and Discussion

Task Performance. The primary purpose of the task in the present study was to place the subject in a situation of task underload, a situation in which boredom would be expected. As would be anticipated on a task of this type, few

error responses were made and typical initial response times were under one second. Two subjects showed decreased response times throughout the test session, and three others maintained relatively steady response times. The remaining seven subjects displayed either slow increases in response time or increased response times during several 5 minute blocks during the test session (typically near the end of the session). At no time did a subject completely fail to respond to the task, although response times exceeding 10 seconds were obtained from two subjects during periods when they appeared to be near sleep.

Heart Rate and Heart Rate Variability. As would be expected from subjects performing a task with low demands, HR (compared for each 5 minute block) showed a slight decrease after the initial block, and then remained steady during the rest of the test session. Despite the relative constancy of average HR, on a within-subjects basis, HRV was found to correlate positively with block number for each of the 12 subjects, indicating an increase in HRV with time on the task. Indices of HRV used in the present study were based on filtering and computing the power spectral density for the IBI data. Two measures of variability were examined: (1) total power, and (2) power in the .05 to .15 Hz bandwidth. This band has typically been associated with blood pressure regulation (Kitney, 1980). Figure 1 shows the plot of power (means for 11 subjects) in the task. The figure shows an increase in average HRV with time on the task.

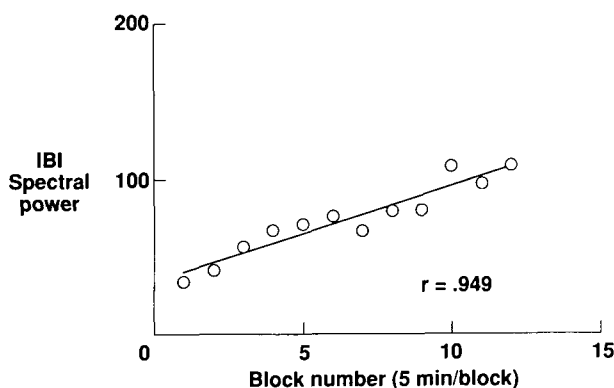


Figure 1. Averaged spectral power (.05 - .15 Hz) of IBI for 11 subjects.

Pupil Diameter. In general, PD has been found to decrease when variety and novelty in the field of view is minimized. Figure 2 presents the relationship between mean pupil diameter for 11 subjects over the course of the test session. As shown in the figure, mean pupil diameter decreased with time on the task.

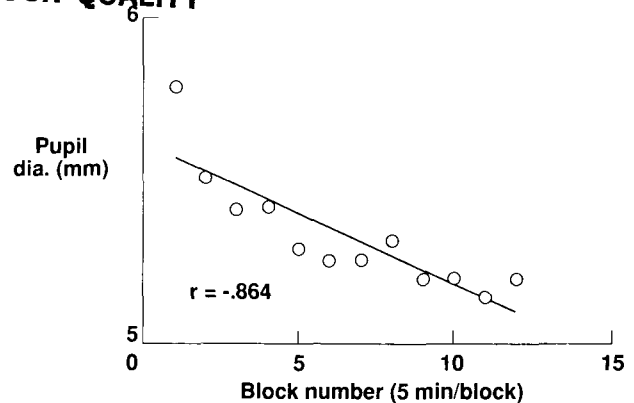


Figure 2. Averaged pupil diameter for 11 subjects.

EEG. As noted above, the EEG signal was filtered into five frequency range components: (1) delta (1-4 Hz), (2) theta (4-8 Hz), (3) alpha (8-13 Hz), (4) beta1 (13-20 Hz), and (5) beta2 (20-40 Hz). EEG data consisted of the level of the integrated amplitude of each of the frequency components (sampled at 1 Hz). An initial hypothesis under test was that continued performance of an underload task would lead to shifts in the EEG activity from faster activity (beta) to slower activity (delta and theta). The data did not support this hypothesis. Analogous to the relationship between HR and HRV, average levels of activity within each frequency band generally remained steady, while variability of the levels within each frequency band exhibited changes with time on the task. Although variability of each

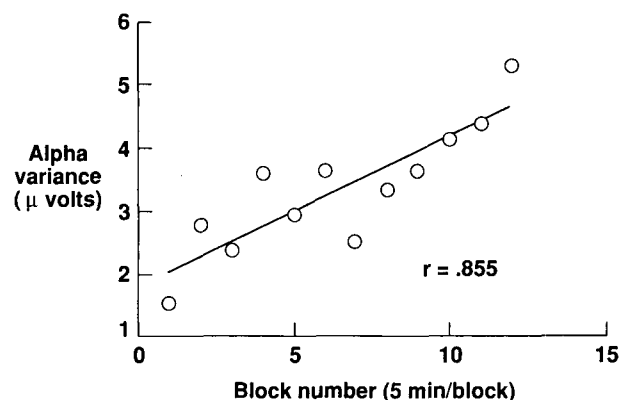


Figure 3. Averaged alpha variance for 11 subjects.

of the lower frequency EEG bands (delta, theta, and alpha) exhibited increases with time on the task, the measure having the highest correlation with HRV and PD was alpha variability. Figure 3 presents alpha variance (means of 11 subjects) versus block number on the task, and shows an increase in alpha variance with time on the task.

Interrelationship of measures. Simultaneous recording of ECG, PD, and EEG permitted examination of the interrelationship between these measures. Figure 4 shows an index of HRV plotted against PD. In this figure the spectral power of the IBI data in the .05 to .15 Hz band (means of 11 subjects) for each block is plotted versus mean pupil diameter for each block. The figure shows increased HRV concomitant with decreased PD. Figure 5 presents the relationship between the HRV index and alpha variance (means of 11 subjects) for each block, and illustrates increases in alpha variance accompanying increases in the HRV index. While the data presented in figures 4 and 5 show high correlation coefficients between measures, when means for the 11 subjects are used, it should be noted that large individual differences in each of the physiological measures are noted when examining the data at the single subject level. Therefore, figures 4 and 5 should not be interpreted simply as suggesting a high degree of convergent validity between measures. Illustrating the independence of the measures, it was found that some subjects displayed changes in one physiological index while showing little or no change in another. These results support the use of profiling techniques in order to establish baselines unique to each individual.

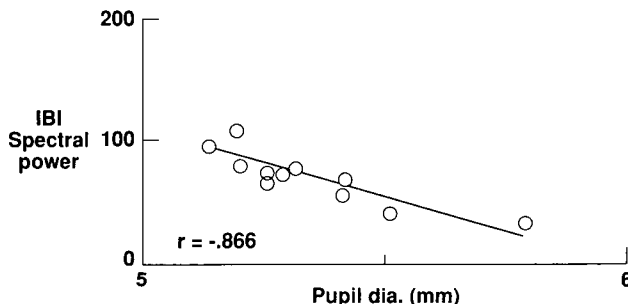


Figure 4. IBI spectral power (.05 - .15 Hz) versus pupil diameter (Mean for 11 subjects).

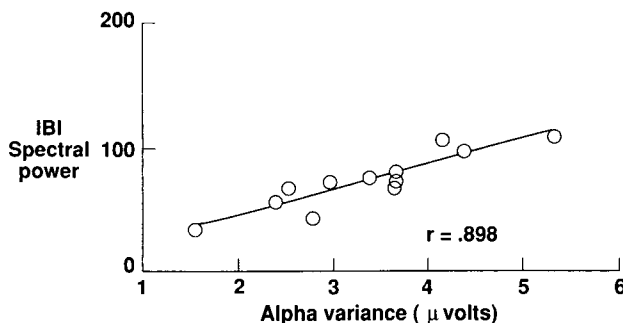


Figure 5. IBI spectral power (.05 - .15 Hz) versus alpha variance (Mean for 11 subjects).

In an underload scenario each of the physiological measures used in the present study are prone to a "ceiling" (or "floor") effect as there are limits to increases in HRV, alpha variability, or decreases in PD. Therefore, it would be expected that over sufficient time periods, these parameters would reach asymptotic levels. As illustrated by figures 1, 2, and 3, which show averages over the test session for HRV, PD, and alpha variance, asymptotic levels were not reached in the one hour test session using the fault acknowledgment task. This suggests that future studies may benefit from a longer period of testing, or perhaps an on-line assessment of physiological indices to determine the duration of an experimental session for a given subject.

Unlike changing the speed of operation or load on a mechanical device, placing a human operator in an underload situation does not guarantee performance deficits or that the operator will be bored. Many of the subjects in the present study reported engaging in a variety of mental tasks or "games" in order to maintain alertness. In effect, the subjects were making changes in load or "busyness" beyond the demands placed on them by the task. Despite engaging in these activities, the majority of subjects exhibited physiological changes during the test session indicative of decreases in arousal level.

The next step in exploring the physiological assessment of task underload is to establish the relationship between observed physiological changes and attributes of task performance. In order to accomplish this step, it is necessary to measure physiological indices while subjects perform (for extended time periods) tasks incorporating and sensitive to the types of activities that operators may encounter in operational settings.

References

- Bergeron, H. P., and Hinton, D. A. (1985). Aircraft automation: The problem of the pilot interface. Aviation, Space, and Environmental Medicine, 56(2), 144-148.
- Chambers, A. B., and Nagel, D. C. (1985). Pilots of the future: Human or computer? Communications of the Association for Computing Machinery, 28, 1187-1199.

ORIGINAL PAGE IS
OF POOR QUALITY

- Cooper, R. (1968). The psychology of boredom. Science Journal, 4(2), 38-42.
- Horst, R. L. (1988). An overview of current approaches and future challenges in physiological monitoring. Mental-State Estimation 1987, NASA CP-2504, 25-42.
- Kitney, R. I. (1980). An analysis of the thermoregulatory influences on heart-rate variability. In Kitney, R. I., and Rempelman, O. (Eds.), The study of heart-rate variability. (p. 81). New York: Oxford University Press.
- Lake, B. W., Suarez, E. C., Schneiderman, N., and Tocci, N. (1985). The Type A behavior pattern, physical fitness, and psychophysiological reactivity. Health Psychology, 4(2), 169-187.
- Mackie, R. R. (1987). Vigilance Research - Are we ready for countermeasures? Human Factors, 29(6), 707-723.
- Mackie, R. R., Wylie, C. D., and Smith, M. J. (1985). Comparative effect of 19 stressors on task performance: Critical literature review (what we appear to know, don't know, and should know). In Proceedings of the Human Factors Society 29th Annual Meeting (pp. 462-469). Santa Monica, CA: Human Factors Society.
- O'Donnell, R. D. (1979). Contributions of psychophysiological techniques to aircraft design and other operational problems. AGARD-AG-244.
- Parasuraman, R. (1987). Human-Computer Monitoring. Human Factors, 29(6), 695-706.
- Perkins, K. A. (1984). Heart rate change in Type A and Type B males as a function of response cost and task difficulty. Psychophysiology, 21(1), 14-21.
- Wiener, E. L. (1985). Beyond the sterile cockpit. Human Factors, 27(1), 75-90.
- Wiener, E. L. (1980). Flight-deck automation: Promises and Problems. NASA TM-81206.

ORIGINAL PAGE IS
OF POOR QUALITYA SCHEMA-BASED MODEL OF SITUATION AWARENESS:
IMPLICATIONS FOR MEASURING SITUATION AWARENESS

Martin L. Fracker, Captain, USAF

Human Engineering Laboratory
Armstrong Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

Measures of pilot situation awareness (SA) are needed in order to know whether new concepts in display design help pilots keep track of rapidly changing tactical situations. In order to measure SA, a theory of situation assessment is needed. In this paper, I summarize such a theory encompassing both a definition of SA and a model of situation assessment. SA is defined as the pilot's knowledge about a zone of interest at a given level of abstraction. Pilots develop this knowledge by sampling data from the environment and matching the sampled data to knowledge structures stored in long-term memory. Matched knowledge structures then provide the pilot's assessment of the situation and serve to guide his attention. A number of cognitive biases that result from the knowledge matching process are discussed, as are implications for partial report measures of situation awareness.

INTRODUCTION

Under the intense stress of combat, military pilots will need to keep track of a rapidly evolving tactical situation. Helping the pilot to maintain his knowledge of the situation from moment to moment, referred to as situation awareness (SA), has become a matter of considerable interest. Measures of pilot SA are needed in order to know whether new concepts in avionics and display design improve SA or not, but psychologists are only now beginning to explore whether and how SA can be measured. Two fundamental questions must be answered before appropriate measures can be developed: precisely what is situation awareness, and how do pilots maintain it. A clear definition of SA is needed because we do not know what to measure otherwise. A model of how pilots maintain SA is needed in order to suggest what kinds of measures will target SA and what kinds will miss the target all together.

In this paper, I will summarize a theory of situation assessment encompassing a definition of SA and a model of how SA is maintained. In the course of this summary, I will show how the theory accounts for certain well-known biases in human cognition.

WHAT IS SITUATION AWARENESS?

In order to define situation awareness, one should first define what a situation is. In this paper, I define a situation to be a set of processes that control events in the environment. At any given moment in time, objects in the environment will be in particular states and at particular spatial locations, but these states and locations are

constantly changing. Therefore, while momentary states of objects are critical to defining a given situation, those states are secondary to the processes that control them. Further, processes in the environment may themselves arise from higher order factors. In combat, for example, there will exist at least two opposing forces, each of which has its own set of goals. In order to achieve those goals, each force will have organized itself in a certain way and will have assigned certain functions to various members of the resulting organization. The processes of combat then arise from the interactions of functions between the two opposing forces.

A situation, then, can be defined at various levels of abstraction. At the highest level, the situation may be defined in terms of the goals of the human participants. At the lowest level, the situation may be defined in terms of the momentary states of objects in the environment. In between these two extremes, the situation may be defined in terms of the organizations, functions, or processes that translate goals into states.

Situation awareness, therefore, can be defined partly as the knowledge that results when attention is allocated to the environment at one or more levels of abstraction. Of course, one could also allocate attention to a particular area within the environment--what Endsley (1988) calls a "zone of interest". Endsley (1988) has defined zones of interest as concentric volumes of space surrounding the pilot throughout which he distributes his attention. But these zones need not be spatially defined. For example, the pilot's own aircraft could define one zone, his own flight could define a larger zone, and the overall battle may define yet a larger zone. Thus, situation awareness should more properly be said to result from the allocation of

attention to a zone of interest at least one level of abstraction. For convenience, I refer to the intersection of zones of interest with levels of abstraction as the "focal region". Assuming that attention is a scarce resource (Kahneman, 1973; Wickens, 1984), situation awareness should be better within a small focal region than within a large one (cf., Eriksen & Yeh, 1985).

A MODEL OF SITUATION ASSESSMENT

Defining situation awareness determines what is to be measured but does not suggest how it should be measured. For this latter purpose, a model of situation assessment is needed. Ideally, such a model will indicate what kinds of measurement operations will target SA and what kinds will miss the target altogether.

Some models of situation assessment stress that pilots develop and maintain a mental representation of the situation in working memory (Endsley, 1988). Because SA is maintained in working memory, these models predict that pilot SA should improve as the pilot's working memory capacity increases. Wickens, Stokes, Barnett, and Davis (1987) have recently provided evidence in support of this prediction. But a strictly increasing monotonic relation between working memory capacity and the quality of SA is expected only if all critical information about the situation must be represented in working memory at all times. This condition would exist only if the environment were the pilot's only source of information. But many theorists propose that recognized patterns among incoming sensory data may identify knowledge structures stored in long-term memory and that these identified structures are also a source of knowledge about the situation (Anderson, 1983; Rumelhart, 1984; Shank, 1982; Wyer & Srull, 1986). The knowledge structures in long-term memory go by different names, depending upon the theorist: associative networks (Anderson, 1983), memory organization packets (Shank, 1982), referent bins (Wyer & Srull, 1986), or schemata (Rumelhart, 1984).

If schemata can provide substantial information about a situation, then the pilot need not attend to every detail of the environment in order to have a reasonably complete assessment of the situation. Rather, he needs to have schemata that accurately fill in many of the details, and he needs to recognize patterns in the incoming sensory data adequate to identify these schemata. Once a schema has been identified, the pilot needs only to search the schema for items of information not currently in working memory.

When an appropriate schema is not found in long-term memory, then the pilot must resort to a backup procedure that greatly increases the load on working memory. This backup procedure has been described by Wyer and Srull (1986). Essentially, the pilot must attend to a larger amount of information in the environment, identify multiple schemata that may be appropriate, place information from these several schemata into working memory, and then integrate the information into a single result.

This model of situation assessment predicts that the relationship between working memory capacity and quality of SA is dependent on the completeness of the knowledge the pilot has stored in long-term memory. If that knowledge is sufficiently complete with respect to a particular focal region, then the quality of SA should be less sensitive to working memory capacity. This dependence on long-term memory suggests that working memory capacity should have a greater impact on the SA of novice pilots than of highly trained expert pilots.

THE MODEL IN OPERATION: COGNITIVE BIASES IN SITUATION ASSESSMENT

Although schemata can facilitate situation assessment and relieve the load on working memory, they can also lead to biases that degrade the quality of situation assessment. These biases are representativeness, availability, the confirmation bias, cue salience, and the "as if" heuristic (see Kahneman, Slovic, & Tversky, 1982; Wickens, 1984; Wickens et al., 1987). These heuristics and biases can be divided into two groups: those that operate when incoming data match some schema, and those that operate when no match is found. Representativeness, availability, and the confirmation bias belong to the first group and are natural consequences of the situation assessment model. Cue salience and the "as if" heuristic belong to the second group and result from the demands of the backup assessment process on limited working memory and attentional resources.

"Representativeness" is defined in Kahneman et al. as the process of matching the pattern of incoming data with a typical pattern for a particular situation stored in long-term memory. Such a matching process is not a computational short-cut as it is sometimes said to be (Wickens et al., 1988) but is instead the central mechanism of situation assessment. Nevertheless, that such pattern matches can sometimes lead to errors in assessment seems indisputable.

One way such matches can go wrong is captured by the availability heuristic.

ORIGINAL PAGE IS OF POOR QUALITY

"Availability" occurs when pilots select the most accessible schema rather than the "best" schema. Within the model, availability results when two or more schemas identify themselves as matching in-coming data and the schema with the strongest level of activation provides the pilot with his situation assessment. Activation strength may be high for several reasons. One is that activation strength should increase as the goodness-of-fit between the data and the schema increases. Another is that a schema may have been primed by earlier events and so already have a high base-line level of activation. If so, then a partial match may result in a higher level of activation than that found in another, unprimed schema where the match was actually better.

The confirmation bias is defined as the tendency to attend only to those sources of information that confirm our previous beliefs. In the present model, the confirmation bias results whenever a schema is activated. The schema directs the pilot's attention to those cues that are relevant assuming that the event represented by the schema is in fact in progress. When the correct schema has been activated, this attentional guidance is beneficial; but if an incorrect schema has been activated, then such guidance can lead to a cascade of assessment errors as one error leads to another.

Cue salience results when activated schemata are not adequate to direct the pilot's attention or when working memory is too overloaded to retain the attentional guidance provided by a schema. In the absence of such attentional guidance, control of the pilot's attention is likely to shift to the external environment. The physical salience of environmental cues may then become the dominant factor guiding the pilot's allocation of attention (cf., Wallsten, 1980).

The so-called "as-if" heuristic also comes into play in the absence of adequate schemata. When incoming data do not match a single schemata, then the data are broken down into subsets and these subsets are then matched to schemata. In the extreme case, each data item in the subset would be matched to a different schema, and the schemata would then serve only to interpret each individual item. The result is that information from multiple schemata will have to be integrated in order to provide a coherent assessment of the situation. In arriving at this assessment, the relative contribution of each item of information should be weighted so that it contributes to the assessment appropriately. But in the absence of a single schema to assign these weights, any weights assigned by the pilot would be arbitrary. Because the simplest set of

arbitrary weights is the set in which all weights are equal, pilots weight each item of information equally. That is, pilots treat the information items "as if" each had the same weight.

IMPLICATIONS OF THE MODEL: MEASURING SITUATION AWARENESS

An important aspect of the model is that once the pilot has achieved an assessment of the situation, that assessment is stored with the schema from which it was derived. If the assessment was integrated from information from multiple schemata, then a new schema is created and stored in long-term memory. At that point, the assessment may no longer be needed in working memory and so may be discarded (see Wyer & Srull, 1986, for a discussion of similar processes). This particular feature of the model has important implications for how SA can be measured, as will now be seen.

Any direct measure of SA will determine what aspects of the situation the pilot has stored in either working or long-term memory. That is, one could ask the pilot for particular kinds of information and then see if he can provide them. Because situation assessments are stored in long-term memory once they have been reached, it will generally not be possible to tell whether the pilot provided the information from working or long-term memory. But because situation assessments may not be retained in working memory once they are no longer needed, it is safest to assume that information is provided from long-term memory. This assumption might seem to suggest that SA could be measured by having pilots recall the details of a mission after the mission had been completed, an approach advocated by Whitaker and Klein (1988) and Kibby (1988). But data on serial position effects suggest that such an approach would measure pilot SA reliably only for events occurring late in the mission (see Tarpy & Mayer, 1978, for a review).

An alternative to post-mission recall is recall during the mission, an approach advocated by Endsley (1988) and Marshak, Kuperman, Ramsey, and Wilson (1987). At various points during the mission, the pilot is asked to report on certain but not all aspects of the mission. For this reason, the approach may be called a partial report procedure. Asking the pilot to recall information about an event during the time that the event is taking place raises certain procedural difficulties. First, the pilot does not need to recall the information if it is currently available in the environment. Therefore, Endsley and Marshak et al. have suggested blanking all displays that might convey the information in question to the pilot. Such a procedure is impossible or at

least extremely dangerous in actual airborne missions and so is used only in simulated missions. Further, to ensure that responding to the memory probe does not interfere with the pilot's mission performance, the simulation is frozen for the probe and then resumed following the pilot's response. A number of practical issues raised by this procedure have been discussed by Endsley (1988) and need not be repeated here.

A theoretical problem with the partial report procedure arises if recall is assumed to be from long-term memory. Essentially, the pilot has to search for the schema in which the relevant assessment was stored, and he has to base this search on the question that was asked. Suppose that the pilot is asked to report the spatial location of a particular enemy aircraft. Now imagine that when the location of that aircraft was noticed, the pilot was trying to determine that aircraft's objectives. Then, the aircraft's location will be stored with a "mission objectives" schema. But the question does not ask for the aircraft objectives, and so the pilot will not search for an objectives schema. As a result, the pilot may be unable to retrieve the aircraft's location even though he noticed and stored it. This theoretical difficulty may not discredit the partial report procedure, but it does suggest that care must be taken in constructing the questions that are to be asked.

Finally, the model suggests that measuring the load on working memory imposed by situation assessment may be as important as measuring SA itself. If pilots attain adequate SA but only at the cost of a high load on working memory, then they would be vulnerable to "losing" their SA if the demands of the mission on working memory were to increase. One way to measure the load on working memory would be by means of a secondary memory span task. Such a task could easily be integrated into the partial report method of measuring SA. Performance on the memory span task in combination with the partial report measure would then provide a measure of the pilot's SA and what it cost him to attain it.

REFERENCES

- Anderson, J. R. (1983). The architecture of cognition. Cambridge, MA: Harvard University Press.
- Brewer, W. F., & Dupree, D. A. (1983). Use of plan schemata in the recall and recognition of goal-directed actions. Journal of Experimental Psychology: Learning, Memory, and Cognition, 9, 117-119.
- Endsley, M. R. (1988). Situation awareness global assessment technique (SAGAT). Paper presented at the National Aerospace and Electronic Conference (NAECON), Dayton, Ohio, May.
- Eriksen, C. W., & Yeh, Y. (1985). Allocation of attention in the visual field. Journal of Experimental Psychology: Human Perception and Performance, 11, 583-597.
- Harwood, K., Barnett, B., & Wickens, C. D. (1988). Situational awareness: A conceptual and methodological framework. In Proceedings of the 11th symposium of psychology in the Department of Defense, April.
- Hastie, R. (1981). Schematic principles in human memory. In E. T. Higgins, C. P. Herman, & M. P. Zanna (Eds.), Social cognition: The Ontario symposium (Vol. 1). Hillsdale, NJ: Earlbaum.
- Kahneman, D. (1973). Attention and effort. Englewood Cliffs, NJ: Prentice-Hall.
- Kahneman, D., Slovic, P., & Tversky, A. (1982). Judgment under uncertainty: Heuristics and biases. London: Cambridge University Press.
- Kibbe, M. P. (in press). Information transfer from intelligent EW displays. In Proceedings of the Human Factors Society 32nd annual meeting. Santa Monica, CA: The Human Factors Society.
- Marshak, W. P., Kuperman, G., Ramsey, E. G., & Wilson, D. (1987). Situation awareness in map displays. In Proceedings of the Human Factors Society 31st annual meeting (pp. 533-538). Santa Monica, CA: The Human Factors Society.
- McKinnon, F. B. (ed.). (1986). Final report of the intraflight command, control, and communications symposium (U). 57th Fighter Weapons Wing, Nellis Air Force Base, NV.
- Pryor, J. B., McDaniel, M. A., & Kott-Russo, T. (1986). The influence of the level of schema abstractness upon the processing of social information. Journal of Experimental Social Psychology, 22, 312-327.
- Rumelhart, D. E. (1984). Schemata and the cognitive system. In R. S. Wyer, Jr., & T. K. Srull (Eds.), Handbook of social cognition (vol 1, pp. 161-188). Hillsdale, NJ: Lawrence Erlbaum.
- Shank, R. C. (1982). Dynamic memory: A theory of reminding in computers and people. Cambridge, England: Cambridge University Press.

**ORIGINAL PAGE IS
OF POOR QUALITY**

Tarpy, R. M., & Mayer, R. E. (1978). Foundations of learning and memory. Glenview IL: Scott Foresman.

Tolk, J. D., & Keether, G. A. (1982). Advanced medium-range air- to-air missile (AMRAAM) operational evaluation (OUE) final report (U). Air Force Test and Evaluation Center, Kirtland Air Force Base, NM.

Vallacher, R. R., & Wegner, D. M. (1987). What do people think they're doing? Action identification and human behavior. Psychological Review, 94, 3-15.

Wachtel, P. L. (1967). Conceptions of broad and narrow attention. Psychological Bulletin, 68, 417-419.

Whitaker, L. A., & Klein, G. A. (1988). Situation awareness in the virtual world: Situation assessment report. In Proceedings of the 11th symposium of psychology in the Department of Defense, April.

Wickens, C. D. (1984). Engineering psychology and human performance. Columbus, OH: Charles E. Merrill.

Wickens, C. D., Stokes, A., Barnett, B., & Davis, T. (1987). Componential analysis of pilot decision making: Final Report (Report No. SCEEE-HER/86-6). Champaign, IL: Aviation Research Laboratory, University of Illinois.

Wyer, R. S., Jr., & Srull, T. K. (1986). Human cognition in its social context Psychological Review, 93, 322-359.

Author's Notes

The author thanks Maris Vikmanis, Mike Vidulich, and Gary Reid who stimulated many of the ideas contained in this paper.

ORIGINAL PAGE IS
OF POOR QUALITYNEXT GENERATION KEYBOARDS:
THE IMPORTANCE OF COGNITIVE COMPATIBILITY

John R. Amell
Computer Sciences Corp.
Cincinnati, Ohio 45201

Michael E. Ewry
Systems Research Laboratories, Inc.
Dayton, Ohio 45324

Herbert A. Colle
Wright State University
Dayton, Ohio 45435

ABSTRACT

The computer keyboard of today is essentially the same as it has been for many years. Few advances have been made in keyboard design even though computer systems in general have made remarkable progress in improvements. This paper discusses the future of keyboards, their competition and compatibility with voice input systems, and possible special-application intelligent keyboards for controlling complex systems.

INTRODUCTION

The keyboards in use today do not differ substantially from those used early in this century. The standard QWERTY keyboard, designed for a wholly mechanical typewriter, has remained unchanged, despite the now totally electronic environment. Substantial amounts of research have been devoted to computer systems, including the man-machine interface, with little improvement over the way data is transferred from human to machine.

Keyboard design research has focused mainly on physical parameters such as height, angle, key resistance, and key shape (Alden, Daniels, & Kanarick, 1972). Key arrangement has also been studied, but very little research has been devoted to innovative keyboard design. Even a supposedly innovative keyboard, the Dvorak keyboard, only focuses on balancing the typing load between fingers and hands. Dvorak rearranged the alphabetic keys based on frequency of use for letters in English text (Dvorak, Merrick, Dealey, & Ford, 1936). Keyboard design has not been directed at capitalizing on the cognitive processes of human operators. Although a considerable amount of research has been directed at the study of cognitive processing during typing, the focus has been to describe the processing underlying the use of QWERTY keyboards rather than being directed at the development of new keyboards (Salthouse, 1986).

When evaluating keyboards to be used for a particular task, several general measures can be used. Accuracy of input, rate of input, and

time to learn combine with physical parameters to provide a measure of keyboard's suitability for a given task. Accuracy and rate of input are straightforward measures. Time to learn a keyboard is dependent upon several variables: the number of possible entries, the type of task, the expected operational input rate and accuracy, and the keyboard's cognitive compatibility. Cognitive compatibility is an important concept, but one that is easy to overlook. A keyboard is more compatible if its functions map regularly into natural human cognitive processes. Keyboards with cognitive compatibility should be easy to learn as well as rapid to operate.

There have been attempts at improving the cognitive compatibility of keyboard designs. These keyboards can be classified as either alphanumeric or icon based. Icon based keyboards include the multi-function keyboards used in fighter aircraft and item selection keyboards used in some fast food restaurants. Chord keyboards, stenotype machines, telegraphs, alphabetic, Dvorak, and QWERTY keyboards are examples of alphanumeric keyboards.

ICON KEYBOARDS

With icon keyboards words, phrases or concepts are processed rather than single letters. Multi-function keyboards, used in fighter aircraft, allow a pilot to select an action, based on the current configuration of the keyboard. Key labels appear on the CRT adjacent to the keys and change to reflect current operability. Other multifunction keyboards have several labels on the key and require the use of shift keys to access all functions. Menu-based computer interfaces are a derivation of an icon keyboard.

Item selection keyboards have dedicated keys, each one corresponds to a particular item. An important characteristic of icon keyboards is that items are organized by class. For example, on a fast food item selection keyboard, sandwiches would be grouped separately from drinks. Within these general classifications, sub-classes also can be grouped. Specialty

sandwiches might be separated from hamburgers within the sandwiches group. This semantic organization is very compatible with operator's cognitive processing when an order is placed. Operators do not have to look up or remember the price of each item. Nor do they have to make several keystrokes to enter the price on the register. (There are other advantages such as communication to team members and inventory which will not be discussed.)

There are, of course, limitations on what can be done with an icon keyboard. The most obvious limitation is that it cannot produce free form text. Also, it is a visual search device which limits the speed at which it can be operated and, typically, it takes considerable space to accommodate all icons.

ALPHANUMERIC KEYBOARDS

Text processing refers to any keyboard used to output letters and words. In alphabetic and QWERTY keyboards each letter has a key that is dedicated to it. A telegraph keyboard is at the opposite extreme. It may have only one key; a pattern of dots and dashes are used to represent individual letters. Chord keyboards are intermediate. With chord keyboards, combinations of several keys are pressed simultaneously to produce letters, syllables, or words.

Alphabetic Keyboards. These keyboards have the keys arranged in alphabetical order, usually in three rows. Basically, the alphabetic keyboard was an attempt to increase cognitive compatibility by taking advantage of a human operator's knowledge of alphabetic order. This attempt has not been successful. Norman (1986) has suggested that the artificial breaks in the alphabet, due to the three rows, reduces the effectiveness of the alphabetic keyboard.

Qwerty Keyboards. Today's microcomputer keyboards, while remaining basically QWERTY text processors, have added function keys, a control (Ctrl) key, and an Alternative (Alt) key. This is a first attempt at integrating both text and concept processing into one keyboard. Typically, Ctrl and Alt key are used to execute a higher level command which would require a string of letters if typed. Often, there has been some attempts to use a mnemonic code to select the command associated with a particular key. Usually this has been to use the first letter of one of the names for the command. If the operator remembers the command name, the result can be higher cognitive compatibility. Seldomly does it achieve the level of compatibility inherent in the use of the shift key. While it is expected that when we strike a "c" on the keyboard, a lower case "c" is produced, and when we strike a "<shift> c", an upper case "C" is the result, it is not so obvious what will result when a "<Ctrl> c" or a "<Alt> c" is pressed. The problem is a lack of cognitive compatibility. The same is true of function keys labeled F1 to F10. There is no inherent meaning in the label.

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z				
0	1	2	3	4	5	6	7	8	9

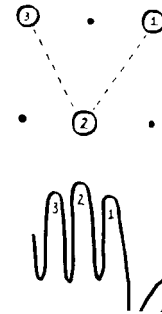


Figure 1 The Alpha-Dot code.

Chord Keyboards. These keyboards typically have fewer keys than standard keyboards since characters are produced by pressing multiple keys on a stroke. Two examples of chord keyboards are the Alpha-Dot keyboard, a one-handed typing device, and the two-handed Stenotype machine. A variety of other chord keyboards have been proposed. See Ewry (1987) for a review.

The Alpha-Dot keyboard has just three character keys and it requires a two strokes for each character entered (Sidersky, 1974). The specially designed character set (Figure 1) visually maps the shape of the printed letter onto a matrix of two rows of three dots, corresponding to the three character keys. Each row of dots represents one keystroke. Tests have shown that people can learn to use this keyboard, by touch, in one hour or less (Amell, 1986; Ewry, 1987). In comparison, when letters are randomly assigned to keystrokes, learning takes considerably longer (Ewry, 1987). Trying to arrange keys based on frequency of use will usually make them appear to be randomly organized. The Alpha-Dot keyboard is learned rapidly because it is cognitively compatible. Compatibility in this case is different than the cognitive compatibility of icon keyboards. The spatial coding scheme of the Alpha-Dot keyboard relates letter shape to key patterns. By using visual imagery, operators can remember many keystroke patterns. A problem with Alpha-Dot coding is that not all letter-keystroke patterns can be represented without resorting to distorted letter shapes.

Stenotypewriters are chord keyboards that have been developed for use in courtroom transcription. These stenotype keyboards use a very different cognitive coding principle. An operator enters a syllable or word with each two-handed keystroke. The machine shorthand code is basically phonetic. Operators enter a consonant-vowel-consonant sequence (CVC) on each stroke, representing a syllable. More complicated CVC patterns also can be entered. Front consonants and back consonants are represented separately on the keyboards. Chord responses are used in two ways. First, to define the CVC. Second, to define consonants and vowels because there are only 22 keys and not all consonant and vowels have a single key definition.

Stenotyping can lead to very rapid text transcription. Expert stenotypists can achieve entry rates of 225-300 words per minute. In contrast, comparable QWERTY typists achieve entry rates of only 60-80 words per minute. Modern online computer transcription can convert stenotype codes into normal alphanumeric text. High entry rates can only be achieved because the system is designed for fast human throughput. On the negative side, however, are the long training times necessary to learn the complete stenotyping code and the high drop out rate of students in training programs. Clearly, stenotyping causes problems for human operators.

Voice Recognition. One solution to the cognitive compatibility problem would seem to be the development of automatic speech recognition. Unfortunately, speech recognition research requires major conceptual breakthroughs before automatic speech recognition can be used routinely. Current systems have trouble segmenting words in continuous speech, and have trouble differentiating homophones. They are also speaker dependent. Currently, most automatic speech recognizers use only a small speaker-dependent vocabulary. Actually, even optimal speech recognition devices would not eliminate the need for keyboards because privacy is required for many keyboard uses. Thus they are similar to icon keyboards.

THE FUTURE OF KEYBOARDS

A small next step would be to integrate what we know about keyboards and speech recognition into a single system. Functions currently accessed by function keys or Alt Ctrl combinations could be called by voice command. This would relieve some of the cognitive incompatibilities present in current systems. Since we generally know what we want to do, we just don't know which key combination does it. This would necessitate the recognition system be part of the front end, even part of the keyboard itself. This would result in personalized keyboards which would be transportable between systems since the output of the keyboard would still be ASCII characters. In addition, since personalized keyboards would have memory for speech recognition, we could also customize the key combinations to produce often used words or phrases instead of functions.

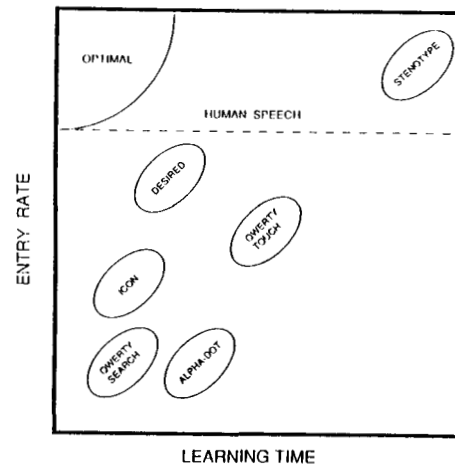


Figure 2 The keyboard design space.

Major changes in keyboard design also may be possible. Figure 2 represents the current state of keyboard design. The axes show the two major parameters of keyboard operation, rate of entry and time to learn. Several keyboards are easy to learn, but their entry rates are low. The stenotype keyboard in the upper right quadrant of the figure represents a keyboard which requires a long learning time, but entry rates are extremely fast. The "desired" ellipse in Figure 2 represents an area of the space where new development work should focus. What is needed is a keyboard in which high entry rates can be achieved with minimal training. By stressing human cognitive processes, this goal may be attainable.

Chord keyboards provide a level of flexibility that lends itself to the development of innovative keyboard designs with high cognitive compatibility. In order to achieve the goal of an easily learned keyboard with entry rates in the range of a QWERTY touch typist or human speech, attention must be paid to the relationship between the operator and the keyboard. One design strategy could be to design down from the stenotypewriter. Simple phonetic coding is easy to learn. Instead of a system for very high entry rates, it may be possible to design a code which is more systematic and regular but at the cost entry rates that are lower than the 225-300 attainable with stenotyping.

Chord keyboards have another advantage. They can be made with very few keys and can be used with one hand. The one-handed chord keyboards may be used as data entry devices where space is a design constraint and speed of entry is noncritical. They could also be used as hand-held remote terminals that communicate with a main computer. The ability to interface with a computer while engaging in activities in the field offers many possibilities. They could be used anywhere a notepad could without the restrictions of even a laptop computer. A keyboard such as the Alpha-Dot could revolutionize laptop computer design.

Keyboards are not going to disappear in the foreseeable future. With current technology, we have the ability to improve their functionality and take greatly improved keyboards into the 21st century.

REFERENCES

- Alden, D.G., Daniels, R.W., and Kanarick, A.F. (1972). Keyboard design and operation: A review of the major issues. Human Factors, 14, 275-293.
- Amell, J.R. (1986). A comparison of two chording systems for alphanumeric data entry. Unpublished master's thesis, Wright State University, Dayton, Ohio.
- Dvorak, A., Merrick, N.L., Dealey, W.L., and Ford, G.C. (1936). Typewriting behavior. New York: American Book Company.
- Ewry, M.E., (1987). Learning chord keyboard codes: A comparison of three coding strategies. Unpublished master's thesis, Wright State University, Dayton, Ohio.
- Norman, D.A. and Fisher, D. (1982). Why alphabetic keyboards are not easy to use: keyboard layout doesn't much matter. Human Factors, 5, 509-519.
- Salthouse, T.A. (1986). Perceptual, cognitive, and motoric aspects of transcription typing. Psychological Bulletin, 99, 303-319.
- Sidorsky, R.C. (1974). Alpha-Dot: A new approach to direct computer entry of battlefield data. Technical Report 249, U.S. Army Research Institute for the Behavioral and Social Sciences.

A METHOD FOR MODELING BIAS IN A PERSON'S
ESTIMATES OF LIKELIHOODS OF EVENTS

Thomas E. Nygren and Osvaldo Morera

Department of Psychology
The Ohio State University
404C W. 17th Avenue
Columbus, OH 43210

It is of practical importance in decision situations involving risk to train individuals to transform uncertainties into subjective probability estimates that are both accurate and unbiased. We have found that in decision situations involving risk, people often introduce subjective bias in their estimation of the likelihoods of events depending on whether the possible outcomes are perceived as being "good" or "bad". Until now, however, the successful measurement of individual differences in the magnitude of such biases has not been attempted. In this paper we illustrate a modification of a procedure originally outlined by Davidson, Suppes, and Siegel [3] to allow for a quantitatively-based methodology for simultaneously estimating an individual's subjective utility and subjective probability functions. The procedure is now an interactive computer-based algorithm, DSS, that allows for the measurement of biases in probability estimation by obtaining independent measures of two subjective probability functions (S^+ and S^-) for "winning" (i.e., good outcomes) and for "losing" (i.e., bad outcomes) respectively for each individual, and for different experimental conditions within individuals. The algorithm and some recent empirical data are described.

It is argued that, if in decision situations involving substantial risk or potential loss, our goal is to train individuals to become expert decision makers, it is important to understand how people subjectively evaluate and represent uncertainties or probabilities. Decision theorists have argued for some time that any decision analysis under risk must involve the assessment of uncertainties, and that uncertainties can best be measured by subjective probabilities that represent the decision maker's degree of belief about the relevant uncertain events. The decision maker must somehow transform uncer-

tainties into subjective probability estimates that are both accurate and unbiased. There is, however, convincing evidence that in many complex decision situations involving risk or uncertainty, people use heuristics that often introduce bias in the subjective estimation of the likelihoods of events that are relevant to the outcomes of the decision [8, 20]. They may judge the probability of an event by its representativeness of a class of events, by its availability in memory as a relevant example, or on the basis of an adjustment from a numerical anchor point.

Recently published edited volumes by Arkes and Hammond [1] and Kahneman, Slovic, and Tversky [10] reflect this new direction of study in the field of judgment and decision making research. Studies have shown that people generally do not make good probability estimates [6, 10, 11, 16, 17, 18]. They overestimate low and underestimate high probabilities and they ignore base-rate information [2, 12]; they revise opinions too conservatively [4]; they indicate excessive confidence in their judgment [7]; and they are influenced by their affective mood state [9, 15].

However, only recently have systematic research efforts investigating the cognitive mechanisms by which biases are generated been reported. For example, Nygren and Isen [15] have shown that a positive mood state can lead decision makers to exhibit "cautious optimism" in risky choice situations. They become optimistic in the sense that they tend to overestimate the likelihood of "good" events and underestimate the likelihood of "bad" events; but at the same time they exhibit a cautious shift toward risk-aversion in their actual choices.

Such findings imply the need for models that interrelate cognitive processes and judgmental biases. Wickens

[21] has argued that without an understanding of these biases in such a framework, it is difficult to predict how specific decisions are being made by individuals. But, how can such biases be quantitatively measured? Most models of decision making under risk assume that there are four basic questions that remain the focal issues in decision analysis. They are: (1) what are the possible courses of action? (2) what are the outcomes associated with these courses of action? (3) what is the utility associated with each outcome?, and (4) what is the probability associated with each outcome? Much quantitative and empirical research continues to focus on Questions 1, 2, and 3, and, in particular, the measurement of *utility* [5, 10]. This paper describes a method to take a closer look at Question 4, the measurement of the *subjective probability function*.

MATHEMATICAL MODELS

There are two leading models of risky decision making upon which this research is based, subjective expected utility (SEU) theory and Kahneman and Tversky's prospect theory [13]. In SEU theory the overall utility of a course of action or "gamble" is found by taking, for each possible outcome in the gamble, the product of the utility of the outcome multiplied by the subjective probability associated with that outcome's occurrence, and summing these terms across all outcomes. The decision maker is assumed to choose the gamble/option with the highest overall expected utility. Prospect theory proposes that the decision process is, in fact, completed in two phases, with the potential courses of action first being "framed" for the choice process. This framing often may constitute a preliminary look at the outcomes, and this look sometimes results in a simplified representation of the choice alternatives, particularly if the alternatives are complex. Following this initial phase, the alternatives are actually evaluated in a manner similar to that suggested by SEU theory, where the alternative with the highest value (utility) is chosen.

Both models take the same general form, then, in that overall preference for a course of action or gamble (G) is assumed to be a function of (a) the values or utilities of the possible outcomes and (b) the subjective probabilities (in SEU theory) or decision weights (in prospect theory) associated with these outcomes. Expressed mathematically, for a simple gamble of the form $G = (x, p; y, 1-p)$ where one obtains outcome x with probability p or

outcome y with probability $1-p$, the subjective value of the gamble (G) is assumed in these models to be

$$V(G) = V(x) * S(p) + V(y) * S(1-p) \quad (1)$$

where $V(x)$ is the utility of outcome x and $S(p)$ is a subjective probability that is associated with outcome, x . In prospect theory $S(p)$ is a *decision weight* rather than a probability estimate, per se. These decision weights are assumed to increase monotonically with objective probabilities of events, but are larger than the objective probabilities for extremely unlikely outcomes and smaller than the objective probabilities for more likely outcomes. In prospect theory the decision weights for complementary events with probabilities p and $1-p$ need not necessarily add to one, but will generally be less than one, a property that Kahneman and Tversky [13] label as *subcertainty*.

However, since both SEU and prospect theory are based on the same simple bisymmetric model, they do not allow for a *differential* weighting of an event's probability in winning versus losing or "good" versus "bad" contexts as we have recently found [9, 15, 16]. That is, the models do not allow for the possibility that a decision maker might weight or even evaluate a probability like .2 or .8 differently, depending on the outcome with which it is associated. To account for such findings one needs a modification of SEU with a *dual* probability function. Such a model has been formally proposed by Luce and Narens [14]. Their dual bilinear model would allow for the measurement of probability bias, where "good" and "bad" outcomes can differentially affect subjective judgments of the same explicitly stated probabilities.

THE QUANTITATIVE METHOD

A modification of the procedure used originally by Davidson, Suppes, and Siegel [3] is now a computer-based algorithm, DSS, (cf. [16]) that independently measures the utility and subjective probability functions (U and S) in Eq. 1 above. Specifically, the S function is measured separately as two functions, S^+ , and S^- , in order to assess potential bias in judgments of the likelihoods of good (S^+) and bad (S^-) outcomes. To the extent that the same function is obtained for S^+ , and S^- , no context bias of this type is present in a decision maker's probability estimates. To the extent that the functions obtained for S^+ , and S^- differ, a *measurable* cognitive bias exists in the individual's probability estimation process.

ORIGINAL PAGE IS OF POOR QUALITY

The DSS procedure involves determining the equality or indifference point in sequences of pairs of gambles so that Eq. 1 can be found for each gamble, and these equations can then be equated and solved in order to estimate the subjective utility associated with various outcomes. This utility function is then used in a second phase to estimate the subjective probability functions. On each trial, an individual is presented with two two-outcome gambles and is asked to indicate which of the two s/he prefers. The two-outcome gambles are set up as follows: Individuals are told that in each gamble, one outcome would be obtained if the event E occurs and the other outcome would be obtained if the event E does not occur. The event E is never specified, but individuals are informed that it has a true computer-generated probability of one-half. (Data from several studies have indicated that such instructions produce no bias between these two alternatives of E and $not E$; individuals indeed weight the two events equally.) On each trial, one gamble, Gamble 1, has both outcomes fixed at specified values (e.g., some amount of money or points); the other gamble, Gamble 2, has one fixed outcome and one that is varied. For each of a specified number of trials (eight in the current version of DSS), the individual is asked to compare Gamble 1 with Gamble 2. The variable outcome in Gamble 2 is modified by DSS contingent upon the individual's response.

The decision maker's task on each trial is simply to indicate which gamble s/he prefers. If Gamble 1 is preferred, the variable outcome in Gamble 2 is adjusted upward by DSS to make this gamble more attractive; if Gamble 2 is preferred, the variable outcome in Gamble 2 is lowered to make this gamble less attractive. The amount of adjustment made by DSS depends on whether the individual indicates that one gamble is either slightly or strongly preferred to the other. Since events E and $not E$ have probabilities fixed at .5 and these events are weighted as equivalent in probability, DSS determines the subjective utility function by noting in the variable-outcome gamble the value/amount necessary for a subject to change his/her preference ordering between the fixed-outcome and variable-outcome gambles (indicating the indifference or equivalence point for that pair of gambles). That is, DSS notes the amount that the individual assigns to the variable outcome in Gamble 2 such that s/he no longer has a clear preference for either Gambles 1 or 2.

One sequence of pairs of gambles we have used with DSS is presented in Table

I; we will use these values throughout the remainder of this paper as an illustration. Each of the eight situations presented in Table I actually consists, then, of a series of adjustments to Gamble 2 that lead to the estimation of a subjective utility scale. For example, for the sequence presented in Table I, in the first situation, the individual is faced with one gamble for which s/he would lose \$10 with $p = .5$ (i.e., if E occurs) and would lose \$10 with $p = .5$ (if $not E$ occurs). This, then, is a sure-loss gamble. The alternative gamble in the pair is described as resulting in a loss of $-\$A$ dollars with $p = .5$, and a gain of \$10 with $p = .5$. The money amount associated with $-\$A$ is initially randomly set to a large negative value or to a large positive value making one gamble initially more attractive. The individual adjusts the variable value up or down as necessary to reach indifference between the gambles ($-\$10$, $-\$10$) and ($-\A, $+\$10$). The final dollar amount associated with $-\$A$ is recorded for the individual by DSS so that this information can be used to determine other utility values in subsequent Trials 3, 4, and 6.

Table I
Construction Sequence for Trials 1 - 8
Used to Find Individual Utility Functions

Trial	Gamble 1		Gamble 2		Subjective Utility
	Get	Get	Get	Get	
1	-10	-10	-A	+10	$V(-A) = -3$
2	+10	+10	-10	+B	$V(+B) = +3$
3	-10	+B	-A	+C	$V(+C) = +5$
4	-A	+10	-D	+B	$V(-D) = -5$
5	-D	+B	-E	+C	$V(-E) = -7$
6	-A	+C	-D	+F	$V(+F) = +7$
7	-D	+F	-E	+G	$V(+G) = +9$
8	-E	+C	-H	+F	$V(-H) = -9$

To facilitate the estimation process, we first assign a utility of +1 to +\$10 and a utility of -1 to -\$10. (Because in SEU theory the subjective utility scale is unique up to an affine transformation, that is, it is interval, we can without any loss of generality assign the utilities of +1 and -1.) Then, after $-\$A$ is found, we can determine that the utility value, $V(-\$A)$, equals -3, by substituting in the formula in Eq. 1

$$S^*(.5) * (-1) + S^*(-.5) * (-1) = S^*(.5) * (+1) + S^*(-.5) * V(-\$A). \quad (2)$$

In a manner comparable to that for Trial 1 in Table I, a value for +\$B can be found next by comparing the gambles (+

\$10, + \$10) and (-\$10, +\$B), yielding the amount that is associated with a utility, $V(+\$B)$, of +3. These values of $-\$A$ and $+\$B$ are then used in Trials 3 - 8 to determine other points on the subjective utility scale. Currently, DSS finds an estimate for $+\$C$, $-\$D$, $-\$E$, $+\$F$, $+\$G$, and $-\$H$, which have utility value of +5, -5, -7, +7, +9, and -9, respectively as shown in Table I.

PROBABILITY ESTIMATION

Once these eight points on the utility function have been obtained along with -1 and +1, a nonlinear regression analysis is completed to find the best fitting utility curve for the estimated point values. A typical observed curve is shown in Figure 1. Given this best fitting curve, other utility values can then be estimated for the individual.

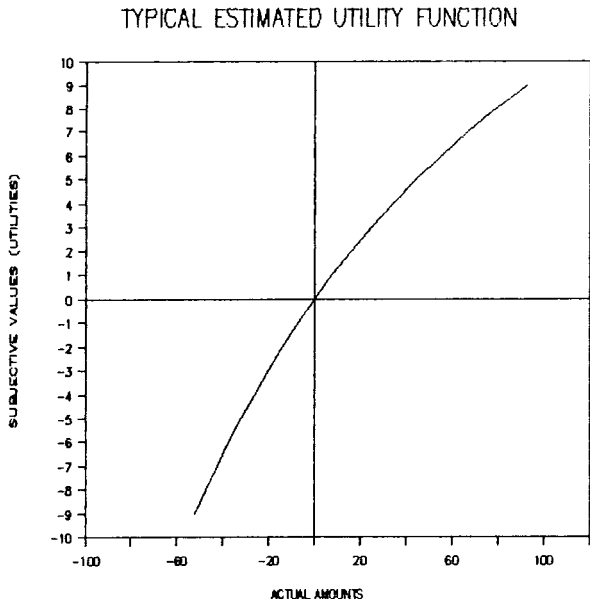


Figure 1. A Typical Utility Curve for Money. Losses End Is Steeper Than Gains End, Indicating Loss-Aversion.

A new series of gambles, now with different explicitly stated probability values, are used to obtain the subjective probability functions for winning (i.e., "good" outcomes) and losing (i.e., "bad" outcomes). Again, in the variable gamble, Gamble 2, one outcome is obtained if the event E ($S(p) = .5$) occurs and the other outcome is obtained if the event E does not occur (also $S(p) = .5$). The fixed gamble is similar to those presented in the utility estimation phase, except that now the probabilities of winning and losing are either .2/.8,

.4/.6, .6/.4, or .8/.2. (Other values could also be programmed into DSS.) The variable outcome in Gamble 2 is again modified in a series of steps until the individual indicates that the two gambles are equally attractive. An example of such a gamble is shown in Figure 2.

GAMBLE PAIR NUMBER 12

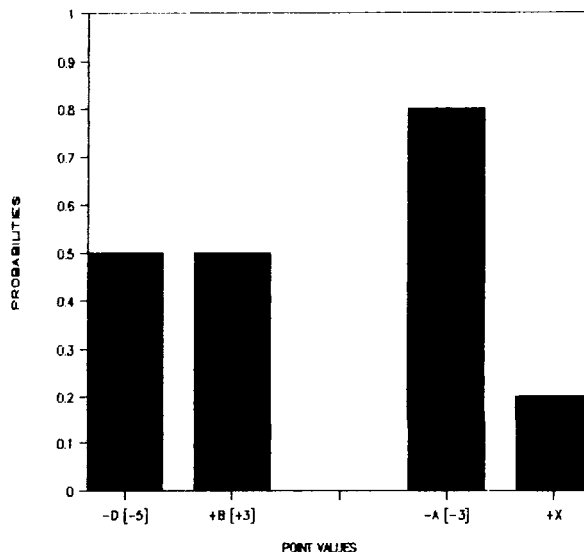


Figure 2. Example of Two Gambles in the Probability Estimation Phase.

Table II shows an illustrative series of eight trials used to estimate the probability functions. Letters A - H represent the amounts found previously on Trials 1 - 8, and X and Y are the variable outcomes; the stated probability values for each outcome are shown in parentheses. Note that on two pairs of trials (Nos. 9/11 and 13/15; Nos. 10/12 and 14/16) a subjective probability of winning value (S^+) and probability of losing value (S^-) is estimated for each of the values 2., .4, .6, and .8.

Table II
Construction Sequence for Trials 9 - 16
Used to Find Individual Probability Functions

Trl	Gamble 1		Gamble 2	
	Get	Get	Get	Get
9	- D(.5)	+ F(.5)	- A(.6)	+ X(.4)
10	- D(.5)	+ B(.5)	- H(.2)	+ X(.8)
11	- A(.5)	+ C(.5)	- D(.4)	+ X(.6)
12	- D(.5)	+ B(.5)	- A(.8)	+ X(.2)
13	- E(.5)	+ C(.5)	- D(.6)	+ Y(.4)
14	- A(.5)	+ C(.5)	- D(.2)	+ Y(.8)
15	- D(.5)	+ B(.5)	- E(.4)	+ Y(.6)
16	- A(.5)	+ C(.5)	-10(.8)	+ Y(.2)

Since events E and *not* E have probabilities fixed at .5 and the events are weighted by the individual as equivalent in probability, we can assign, with loss of generality, $S^*(.5) = S^-(.5) = .5$. Equations 3 and 4 illustrate two examples of trials where $S^*(.8)$ and $S^-(.2)$ are then determined as follows:

$$\begin{aligned} S^*(.5)*V(+ \$B) + S^-(.5)*V(- \$D) = \\ S^*(.8)*V(+ \$X) + S^-(.2)*V(- \$H) \end{aligned} \quad (3)$$

$$\begin{aligned} S^*(.5)*V(+ \$C) + S^-(.5)*V(- \$A) = \\ S^*(.8)*V(+ \$Y) + S^-(.2)*V(- \$D) \end{aligned} \quad (4)$$

If we assume that these two subjective probabilities act like objective probabilities and add to one, where $S^*(.8) = 1 - S^-(.2)$, then by substituting in utility values obtained from the estimated utility curve for $-\$A$, $+\$B$, $+\$C$, $-\$D$, $-\$H$, $+\$X$, and $+\$Y$, two independent estimates of $S^*(.8)$ and $S^-(.2)$ can be obtained by solving Eqs. 3 and 4. If we do not wish to assume that these two subjective probabilities add to one, we can still estimate $S^*(.8)$ and $S^-(.2)$ since we have two equations and two unknowns. In a comparable manner we can find restricted (i.e., add to one) or unrestricted (i.e., do not add to one) estimates of $S^-(.8)$ and $S^*(.2)$ from two new equations, and similarly for $S^*(.6)$ and $S^-(.4)$ and $S^-(.6)$ and $S^*(.4)$. Other probability values could also easily be estimated.

Regardless of whether a restricted or unrestricted model is fit, and regardless of the shape of each individual's subjective utility function, the S^+ and S^- functions should be the same if no probability bias due to a good vs. bad context exists in an individual's data. That is, if no bias exists, we should expect $S^-(.2) = S^*(.2)$, $S^-(.4) = S^*(.4)$, $S^-(.6) = S^*(.6)$, and $S^-(.8) = S^*(.8)$.

AN EMPIRICAL EXAMPLE

Subjects. Thirty-two male and female undergraduate students at The Ohio State University volunteered to participate in this study in partial fulfillment of the requirements of their introductory psychology course.

Procedure. On each trial, subjects were presented with two two-outcome gambles and were asked to indicate which of the two they preferred. In this way both subjective utilities and subjective probabilities for .2 - .8 were independently estimated. The subjects were instructed that they would be asked to make choices between pairs of gambles, and after they had indicated all of their preferences, some of the choice situations would be randomly sampled and

played. The gamble that they would play in each pair would be randomly picked between the two gambles indicated to be equally attractive on each trial. Each subject was given 100 points, representing his or her credit for participating and was told that s/he would be gambling with this credit (not money) in the randomly selected gambles. Subjects were told that as a result of the gambling, they might either lose their credit hour if they lost their 100 points, retain their credit hour if they finished with more than zero points, or gain an additional credit if they finished with more than 200 points. At this point, all subjects were given the opportunity to withdraw from the study without penalty. None did so.

Results. Table III presents the estimated subjective probabilities that were obtained for .2, .4, .6, and .8 for both the winning and losing contexts. The data labeled "restricted" represents, as discussed above, the average of two estimates that are based on the restriction that the estimates for complementary events sum to one. The data labeled "unrestricted" are based on the estimates found when the sums are not restricted to add to one. Regardless of which of these models is assumed, the estimates in the table indicate that indeed across individuals a strong biasing effect exists. The same objectively stated probability values (.2, .4, .6, and .8) when presented to individuals, elicit consistently different subjective values or weights in their decision making process. At all four estimated levels of probability, the estimates that were associated with winning outcomes were consistently weighted lower than the corresponding values for losing. This represents a very strong affective bias in the estimation and/or weighting processes for subjective likelihoods.

Table III
Median Estimates of Winning and Losing
Subjective Probabilities

Actual	Restricted Case		Difference
	Prob L	Prob W	
.200	.347	.208	.139
.400	.462	.401	.061
.600	.599	.539	.060
.800	.793	.653	.040
Actual	Unrestricted Case		Difference
	Prob L	Prob W	
.200	.312	.238	.074
.400	.469	.400	.069
.600	.567	.508	.059
.800	.815	.514	.301

SUMMARY

When an event has an established probability associated with it, it should be irrelevant whether that event is associated with a "good" or "bad" outcome context; the affective nature of the outcomes should not influence probability estimation or weighting in the decision process. In this study, two probability functions were estimated, and different values were found to be assigned to the winning and losing events. This differential weighting of the same event suggests that there is an affective influence on probability estimation in the decision making process. Individuals make choices between alternatives by assigning different subjective probabilities or weights to the same explicit event depending on whether it has a positive affective component (winning) or a negative affective component (losing).

The DSS methodology is important because it has the potential of not only quantitatively measuring this probability bias but also of explaining how some biases in probability estimation may cause suboptimal decisions, and how such bias can be reduced or eliminated in training decision makers to become more "expert" judges. It is designed to lead to programmatic research that has the ultimate application of developing training procedures that can: (a) standardize probability estimation methods in decision making under risk, (b) eliminate estimation biases such as over- and underestimation, (c) reduce individual differences in probability estimation, and (d) develop a scale for assessing a decision maker's accuracy and unbiasedness in subjective probability estimation. Research in this area is necessary if we are to go beyond merely describing suboptimal decision making behavior. The present study is an attempt to begin programmatic research that will allow us to predict suboptimal behavior due to biases in probability estimates and to train individuals to reduce bias in their judgments. In particular, several issues seem to be initially relevant to continued research. First, how strong and how generalizable is the differential weighting effect for probabilities? Second, what factors influence the strength of this effect? And, third, can a method like the DSS procedure coupled with the dual bilinear model allow us to predict and quantitatively measure the effects of suboptimal decision making strategies?

The research reported here, together with that reported elsewhere [8, 9, 10, 15, 16] is resulting in a more complete picture of the complex role that

estimation bias and affect play in decision-making under risk or uncertainty. Our findings suggest that the dual bilinear model is a model worth pursuing. It has the potential to explain a number of difficult findings in the decision making literature including the framing effect [13], the differential weighting effect [16], and the "cautious optimists" effect [15]. Finally, the DSS procedure offers a quantitative measurement procedure for actually measuring rather than simply describing biases in judgment and decision making processes.

ACKNOWLEDGMENTS

This research was supported in part by Contract F33615-85-D-0514 from the U. S. Air Force and Grant S-760-7MG-052 from the Air Force Office of Scientific Research, Bolling AFB, DC to the first author.

REFERENCES

1. Arkes, H. R., and Hammond, K. R., (Eds.), *Judgment and decision making: An interdisciplinary reader*, (Cambridge University Press, New York, 1986).
2. Bar-Hill, M., "The base rate fallacy in probability judgments," *Acta Psychologica*, 44, 1980, 211-233.
3. Davidson, D., Suppes, P., & Siegel, S., *Decision making: An experimental approach*, (Stanford University Press, Stanford, CA, 1957).
4. Edwards, W., "Conservatism in human information processing," in: B. Kleinmuntz (Ed.), *Formal representation of human judgment*, (Wiley, New York, 1968), 17-52.
5. Einhorn, H. J., & Hogarth, R. M., "Behavioral decision theory: Processes of judgment and choice," *Annual Review of Psychology*, 32, 1981, 53-88.
6. Fischhoff, B. "Perceived informativeness of facts," *Journal of Experimental Psychology: Human Perception and Performance*, 3, 1977, 349-358.
7. Fischhoff, B., & Slovic, P., "A little learning ...: Confidence in multi-cue judgment tasks," in: R. E. Nickerson (Ed.), *Attention and performance (Vol. 8)*, (Erlbaum, Hillsdale, NJ, 1980), pp. 779-800.

8. Hogarth, R., "Beyond discrete biases: Functional and dysfunctional aspects of judgmental heuristics," *Psychological Bulletin*, 90, 1981, 197-217.
9. Isen, A. M., & Nygren T. E., "The influence of positive affect on the subjective utility of gains and losses: It's just not worth the risk," *Journal of Personality and Social Psychology*, in press, 1988.
10. Kahneman, D., Slovic, P., & Tversky, A., (Eds.), *Judgment under uncertainty: Heuristic and biases*, (Cambridge University Press, New York, 1982).
11. Kahneman, D., & Tversky, A., "Subjective probability: A judgment of representativeness," *Cognitive Psychology*, 3, 1972, 430-454.
12. Kahneman, D., & Tversky, A., "On the psychology of prediction," *Psychological Review*, 80, 1973, 237-251.
13. Kahneman, D., & Tversky, A., "Prospect theory: An analysis of decisions under risk," *Econometrica*, 47, 1979, 263-291.
14. Luce, R. D., & Narens, L., "Classification of concatenation measurement: Structures according to scale type," *Journal of Mathematical Psychology*, 29, 1986, 1-72.
15. Nygren, T. E., & Isen, A. M., "Examining probability estimation: Evidence for dual subjective probability functions," Paper presented at the Psychonomic Society Meetings, Boston, MA, November, 1985.
16. Nygren, T. E., & Morera, O., "Davidson, Suppes, & Siegel revisited: Evidence for a dual bilinear model," Paper presented at the Mathematical Psychology Meeting, Evanston, IL, July, 1988.
17. Slovic, P., Fischhoff, B., & Lichtenstein, S., "Facts and fears: Understanding perceived risk," in: R. Schwing & W. A. Albers, Jr. (Eds.), *Societal risk assessment: How safe is safe enough?* (Plenum Press, New York, 1980) Plenum., 181-216.
18. Slovic, P., Fischhoff, B., & Lichtenstein, S., "Response mode, framing, and information-processing effects in risk assessment," in: R. Hogarth (Ed.), *New directions for methodology of social and behavioral science: No. 11, Question framing and response consistency*, (Jossey-Bass, San Francisco), 1982, 21-36.
19. Tversky, A., & Kahneman, D., "Availability: A heuristic for judging frequency and probability," *Cognitive Psychology*, 5, 1973, 207-232.
20. Tversky, A., & Kahneman, D., "Judgment under uncertainty: Heuristics and biases," *Science*, 185, 1974, 1124-1131.
21. Wickens, C. D., Stokes, A., Barnett, B., & Davis, T. "Componential analysis of pilot decision making," USAF Technical Report, June 1987.

AN INTELLIGENT MULTI-MEDIA HUMAN-COMPUTER DIALOGUE SYSTEM

J.G. Neal, K.E. Bettinger, J.S. Byoun, Z. Dobes, & C.Y. Thielman
 Calspan-UB Research Center (CUBRC)
 4455 Genesee Street
 Buffalo, New York 14225

ABSTRACT

Sophisticated computer systems are being developed to assist in the human decision-making process for very complex tasks performed under stressful conditions. The human-computer interface is a critical factor in these systems. The human-computer interface should be simple and natural to use, require a minimal learning period, assist the user in accomplishing his task(s) with a minimum of distraction, present output in a form that best conveys information to the user, and reduce cognitive load for the user. In pursuit of this ideal, the Intelligent Multi-Media Interfaces project, sponsored by the Defense Advanced Research Projects Agency and monitored by the Rome Air Development Center, is devoted to the development of interface technology that integrates speech, natural language text, graphics, and pointing gestures for human-computer dialogues. The objective of the project is to develop interface technology that uses the media/modalities intelligently in a flexible, context-sensitive, and highly integrated manner modelled after the manner in which humans converse in simultaneous coordinated multiple modalities. As part of the project, a knowledge-based interface system, called CUBRICON (CUBRC Intelligent CONversationalist) is being developed as a research prototype. The application domain being used to drive the research is that of military tactical air control.

1. INTRODUCTION

As the number and sophistication of military information processing systems rapidly increases, the impact on human operational users must be considered very carefully. Typically, large amounts of information must be communicated for use by the human operator in performing time-critical decision-making tasks for command and control functions. The problem is to make such sophisticated systems easy for military operators to use quickly and efficiently. Modern information processing and decision-aiding systems require a full range of communication media to facilitate interaction and provide the increased bandwidth for information transfer with the human user. The human-computer interface must not only take advantage of multiple media/modalities, but must make use of the synergistic properties of these media to minimize the user's cogni-

tive workload: (1) the human-computer interface system must provide the user with input media/modalities that are natural and efficient for the user; and (2) the use of multiple media/modalities must be applied to the problem of presenting output information to the user in a manner that maximizes user comprehension; the manner of presentation must be based on knowledge of the application domain, the characteristics of the information, the discourse context, the user's task, and respected human factors guidelines.

Presently, there is no computer system that can meet the above requirements. Knowledge-based understanding and generation of information by a computer system in multiple media/modalities has recently begun to be investigated [Neches & Kaczmarek, 1986; Kobsa et al., 1986; Arens et al., 1988; Neal & Shapiro, 1988; Neal et al., 1988; Sullivan & Tyler, 1988]. The research discussed in this paper is part of the Intelligent Multi-Media Interfaces (IMMI) project [Neal & Shapiro, 1988; Neal et al., 1988] which is dedicated to the development of intelligent multi-media interface technology that integrates speech, natural language text, graphics, and pointing gestures for human-computer dialogues. The objective of the project is to develop interface technology that uses the media/modalities intelligently in a flexible, context-sensitive, and highly integrated manner modelled after the way in which humans converse in simultaneous coordinated multiple modalities. As part of the project, a knowledge-based interface system, called CUBRICON (the CUBRC Intelligent CONversationalist), is being developed as a proof-of-concept prototype. Although the IMMI project is a basic research project, an application task domain of military tactical air control is being used to drive the research.

This paper discusses the CUBRICON human-computer dialogue system and the multi-media communication methodology which forms the foundation of the system. Section 2 presents a brief overview of the CUBRICON system. Section 3 discusses the knowledge sources used by the system. Section 4 focuses on multi-media input processing while Section 5 focuses on the generation of multi-media output. Subsequent sections outline the future direction for the CUBRICON project, summarize the main ideas of this paper, provide acknowledgements, and list references.

2. OVERVIEW

The model upon which the CUBRICON system is based is that of two people talking in front of a blackboard or other graphics display device(s). People in such a situation use various combinations of modalities for very effective communication. The modali-

This research is supported by the Defense Advanced Research Projects Agency and monitored by the Rome Air Development Center under Contract No. F30603-87-C-0136.

ties include spoken natural language, written text, pointing gestures, drawings, and tables. Thus, CUBRICON is intended to imitate, to a certain extent, the ability of humans to simultaneously accept input from different sensory devices (such as eyes and ears), and to simultaneously produce output in different media (such as voice, pointing motions, and drawings).

CUBRICON accepts input from three input devices: speech input device, keyboard, and mouse device pointing to objects on a graphics display. Output is produced for three output devices: color-graphics display, monochrome display, and speech output device. The CUBRICON software is implemented on a Symbolics Lisp Machine using the SNePS semantic network processing system [Shapiro, 1979; Shapiro & Rapaport, 1986], an ATN parser/generator [Shapiro, 1982] and Common Lisp. Speech recognition is handled by a Dragon Systems VoiceScribe 1000. Speech output is produced by a DECTalk speech production system.

Every intelligent entity requires a considerable amount of knowledge upon which to base its decision-making processes. Certain knowledge sources have been identified as essential to CUBRICON's multi-media communication ability. These knowledge sources are discussed in the next section.

3. KNOWLEDGE SOURCES

In order for CUBRICON to perform the critical functions of a human-computer interface, the following knowledge sources were found to be essential: (1) domain-specific and related interface knowledge, (2) multi-media language knowledge, (3) the discourse context, (4) the user/task model, and (5) human factors guidelines for enhancing human understanding, and (6) information characteristics.

3.1 Domain-Specific And Related Interface Knowledge

CUBRICON includes a knowledge base containing domain-specific and interface information. The domain-specific information is applicable to the mission planning task domain of the target application system. Task domain entities include airbases, surface-to-air missile (SAM) systems, fuel storage facilities, and targets. The knowledge base also includes essential information concerning how to present or express the various entities via the system's verbal/graphic language. This information includes the words and symbols used to express any given entity, which symbols are appropriate under which conditions, and when particular colors are to be used.

3.2 Multi-Media Language Knowledge

CUBRICON's multi-media language is defined by the combination of its lexicon and grammar. A lexicon is the collection of all morphemes, tokens, or signals that carry meaning in a given language. CUBRICON's lexicon consists of words, graphic figures, and pointing signals. The grammar defines how the morphemes, tokens, and signals of the lexicon can combine to form legal composite language structures. An example of a multi-media language structure that is legal according to the CUBRICON grammar is a noun phrase consisting of the typical linguistic syntax, accompanied by one or more pointing signals (pointing to objects on a graphics display).

3.3 The Discourse Model

The attentional discourse focus space [Grosz, 1978, 1986; Sidner, 1983; Grosz and Sidner, 1985] is a key knowledge structure that supports continuity and relevance in dialogue. The CUBRICON system tracks the attentional discourse focus space of the dialogue carried out in multi-media language and maintains a representation of the focus space in two structures: (1) a main focus list and (2) a set of ancillary focus lists called virtual displays. The main focus list includes those entities and propositions that have been explicitly expressed (by the user or by CUBRICON) via natural language, pointing, highlighting, or blinking. A virtual display is a list of all the objects that are "in focus" because they are visible in a given window on one of the displays. CUBRICON maintains one virtual display per window.

3.4 User/Task Model

A user/task model is essential as a basis for judging the relevance and importance of information items to the user. Carberry [1987] provides a brief summary of current research on user modeling. CUBRICON's user/task model includes (1) a task hierarchy and (2) an entity rating module.

The task hierarchy is a decomposition of the user's main tasks into subtasks. In a task oriented application system, there is usually some a priori knowledge of the task hierarchy and sequencing. Even though the task hierarchy structure is not absolute in that a user may deviate from the typical roadmap through the tasks, this hierarchy can be used as a valuable knowledge source to track the discourse focus, manage the displays, and anticipate the needs of the user.

The CUBRICON entity rating module includes a task-dependent representation of the relative importance of all the entity types known to the system and an algorithm for modifying these ratings depending on task and discourse context. CUBRICON uses a numerical value (on a scale from 0.0 to 1.0) to represent an entity's importance. A pre-defined task-dependent initial assignment of ratings to entities is used when a new task is started by the user. These ratings are modified when the entities are referenced in the dialogue.

3.5 Characteristics Of The Information

The characteristics of the information to be expressed are critical to the selection of an appropriate presentation modality. The following list briefly summarizes CUBRICON's criteria for selecting presentation modality based on characteristics of the information. The CUBRICON design is based on the premise that graphic/pictorial presentation is always desirable.

- o Color-graphics: Selected whenever CUBRICON knows how to represent the information pictorially.
- o Table: Selected when the values of common attribute(s) of several entities must be expressed.
- o Histogram: Selected when a quantitative attribute of several entities must be displayed in a comparative form.
- o Form: A predefined form is selected when the task engaged in by the user requires the form.
- o Printed natural language: Selected for the expression of a proposition, relation, event, or combination thereof that would strain the user's short term memory if speech were used (e.g., technical responses that must be referred to subsequently by the

user); the knowledge structures being expressed are heterogeneous;

- o Spoken natural language: Selected for explanations of displays, verbal highlighting of objects on the displays, informing the user about the system's activity (e.g., "I'm still working" when the user must wait for output from the system), short expressions of relatively non-technical information that can be remembered when presented serially.

3.6 Guidelines For Enhancing Human Understanding

Guidelines from the human factors and psychology disciplines for enhancing human understanding are being incorporated into the CUBRICON system. Smith & Mosier [1986] provide one of the well-known sources of such human factors guidelines. Guidelines that have been incorporated in the CUBRICON system include:

- o Maintain the context of the user/computer dialogue. For example, in managing map displays, CUBRICON tries to retain the most recently discussed or mentioned objects on the displays (avoid premature removal) so as to maintain continuity in the dialogue.
- o Maintain consistency throughout a display. For example, when CUBRICON expands or extends a map display to include some new area, then the same types of objects are displayed in the new region as in the old so that the user does not incorrectly infer that a certain type of object is *not* in the newly added region because it is not being displayed.
- o Maintain consistency across displays. Different displays of the same type should have the same general format. Color keys, titles, help information, etc., should be consistently located for different displays of the same type.

4. MULTI-MEDIA INPUT UNDERSTANDING

People commonly and naturally use coordinated simultaneous natural language (NL) and pointing gestures. These two modes of communication combine synergistically to form an efficient method of expressing definite references and locative adverbials. For example, a person could simply say "this SAM" (surface-to-air missile) and point to an entity on the display to select from among several SAM systems visible on the display. Used in isolation, each of the two modes have shortcomings. If natural language is used alone, then lengthy descriptions are frequently required to identify objects that lack unique "names". For example, to specify a particular SAM from among many SAMs visible on a map display, a person would need say something like "the SAM system at 12.3 degrees longitude and 50.5 degrees latitude" or "the SAM system just east of Kleinburg".

Pointing used alone also has problems: (1) a point gesture can be ambiguous if the point touches the area where two or more graphical figures or icons overlap and (2) the user may inadvertently miss the object at which he intended to point. This latter problem is discussed briefly in the last paragraph of this section. To handle the first problem of ambiguous pointing, some systems use default techniques. These techniques include: (1) a point returns the entity represented by the "top" or "foremost" icon where the system has a data structure it uses to remember the order in which icons are "painted" on the display (i.e., which are further in the background

and which are foremost in the foreground); (2) the icons or entities are assigned weights representing importance and the icon with the largest weight is selected as the interpretation of an ambiguous point; or (3) the icon whose "center" is closest to the location pointed at is selected. Combinations of such techniques can also be used. A serious disadvantage of the above listed point-interpretation techniques is that it is difficult, if not impossible, for certain icons to be selected via a point reference.

CUBRICON's acceptance of dual-media input (NL accompanied by coordinated pointing gestures) overcomes the limitations of the above weak default techniques and provides an efficient expressive referencing capability. The CUBRICON methodology for handling dual-media input is a decision-making process that depends on a variety of factors such as the *types* of candidate objects being referenced, their *properties*, the *sentential context*, and the *constraints on the participants* or fillers of the semantic case frame for the verb of any given sentence. CUBRICON's decision-making process draws upon its knowledge sources discussed in Section 3.

CUBRICON also provides the user with flexibility in several different ways: (1) the user can input natural language via either the speech device or the keyboard, (2) the user is not limited to just one point per NL phrase, but can point several times per phrase, (3) point gestures can occur anywhere within a given NL phrase, and (4) the user can reference four different types of objects via pointing: geometric points, entities represented graphically (e.g., by icons), entries in table displays, and windows on a display.

We present a few brief examples to illustrate the functionality of the system. In each of the following examples, assume that the <point> touches one or more icons.

Example 1: USER: "What is the status of this <point> airbase?"

From the icons touched by the point, the virtual display is searched for the semantic representation of the objects which were graphically displayed by the touched icons. From the hierarchy of the knowledge base, the system determines which of the objects selected by the point gesture are airbases and discards the others.

Example 2: USER: "What is the mobility of this <point>?"

Example 2 entails the use of the property "mobility" as the critical item of information that is used to determine the referent of the dual-media phrase. After searching the virtual display for the objects touched by the point gesture, CUBRICON determines which of these objects have property "mobility" using the knowledge base of application information.

Example 3: USER: "Remove this window <point>."

In example 3, assuming the <point> touches an icon on a window of the display, the term "window" used with the point gesture insures that the system correctly interprets the reference, which would otherwise be ambiguous.

CUBRICON also includes the ability to handle two types of "ill-formed" input: dual-media expressions which (1) are inconsistent or (2) have an apparent null referent. A dual-media expression is inconsistent when the natural language part of the expression and the accompanying point cannot be interpreted as referring to the same object(s) (e.g., the user says "this airbase" and points to a factory). A dual-media expression has no apparent referent when the

user's point touches no icons (i.e. he points to an "empty" area). In both of these cases, CUBRICON infers the intended referent. The CUBRICON methodology for processing multi-media input is discussed in greater detail in [Neal et al., 1988].

5. MULTI-MEDIA OUTPUT GENERATION

The CUBRICON system is being developed so that it embodies the following key features which are essential to maximizing human understanding of presented information: (1) Output presentations should be sensitive to context and relevance. (2) Selection of appropriate presentation media/modalities should be based on the characteristics of the information to be expressed, alternatives being selected when necessary. (3) The media/modalities should be used in a highly integrated manner during output presentation. (4) Respected human factors guidelines should be adhered to.

The CUBRICON output planning process is highly dependent on the knowledge sources discussed in Section 3. The top level output planning process is summarized below.

- 1. For each information item or cluster, determine the modality in which it should ideally be expressed. Graphic/pictorial presentation is always desirable. Natural language can always be used as a last resort.
- 2. Determine whether the resources are available to express the information as desired. Resources: (1) Color graphics display: Are the items to be expressed graphically already on the display? If so, no additions are necessary. If not, is there room to insert them in their "natural" position? (2) Monochrome display: Similar to the color graphics display. (3) Natural language text window: Always available due to scrolling capability. (4) Speech output device: Always available.
- 3. If the desired display cannot accommodate the new information without modification, determine whether the state of the display can be modified. This depends on the level of importance of the information to be expressed and the task model.

Possible modifications to the graphics display:

- o Extend the displayed region to include information items to be expressed.
 - o "Zoom in" to display a portion of the current display in greater detail.
 - o "Pan" to a different region.
 - o Combination of the above.
 - o Open a window on the display to show the new information.
- 4. If the display status cannot be adequately modified as per step 3 above, try modifying the information to be expressed: trim the amount of information by filtering on the basis of relevance with regard to the user/task model and/or the discourse model.
 - 5. If the information can still not be expressed in the given modality due to insufficient resources for the selected modality, then select another modality and go back to step 2.
 - 6. Finish composing the output having resolved resource restraints.

The following are working examples from a dialogue with CUBRICON to illustrate the functionality of the system. These examples illustrate the output composition process and use of the knowledge sources discussed in the previous sections. The dialogues are concerned with mission planning and situation assessment in a tactical air control domain.

USER: "Display the Inner Fulda Gap Region."

CUBRICON:
Color Graphics Display:

- o Map of Inner Fulda Gap Region with main roads, major cities, waterways, and national boundaries (Figure 1).
- o Icons representing entities within the Inner Fulda Gap Region that are above a preset threshold in importance superimposed on the map (Figure 1).

Entities in the Region					
Item	Disposition	Latitude	Longitude	Name	Mobility
fighter base	enemy	10 4100	10 5400	airbase	---
fighter base	enemy	11 4000	11 4000	airbase	---
SA-2	enemy	10 5100	10 5100	---	low
SA-2	enemy	10 4900	11 0000	---	low
SA-1	enemy	10 4900	10 5000	---	high
SA-1	enemy	11 4000	11 1100	---	high
steel plant	friendly	11 4100	9 5400	steelmill steel	---
munition factory	friendly	49 5100	10 1100	factory munitions	---

Display the Fulda region.
11

Mon 11 Jul 81 06:22 paul@tict CL SHEPS: user input JCR

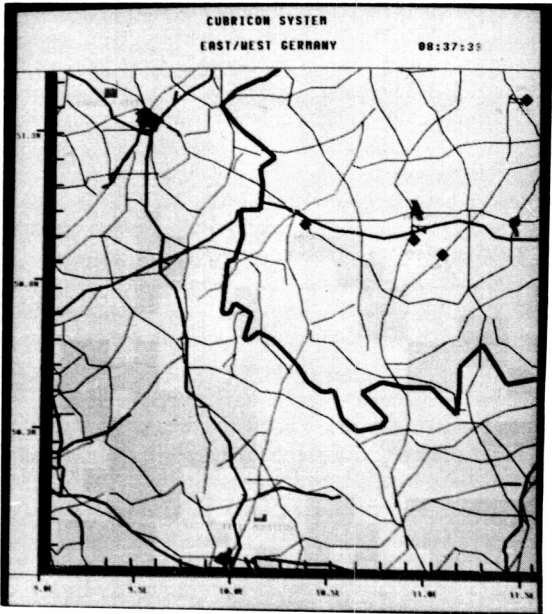


Figure 1. Multi-Modal Presentation Composed by CUBRICON

ORIGINAL PAGE IS OF POOR QUALITY

Monochrome Graphics Display:

- o Table of relevant attributes for the entities that are displayed on the map display.

DISCUSSION:

CUBRICON's first choice for displaying information is graphically, when appropriate. The Fulda Gap Region boundary is defined in the CUBRICON knowledge base and provides sufficient information for the modality selection process. CUBRICON selects color-graphics as a display modality and generates a map display. Since this was the initial input, the color graphics display was available.

An important feature of CUBRICON is that output presentations are formulated in a context sensitive manner. Entities to be displayed on the map are selected on the basis of their importance to the task at hand. Only entities above a pre-set threshold on the entity rating scale are displayed. Non-domain-specific items such as national borders, roads, rivers, and cities are displayed using the MAP Display System [Hilton, 1987].

Another key feature of the CUBRICON interface system is that it is multi-media. In the present example, the CUBRICON output uses both color graphics (i.e., the map display), and a tabular presentation on the monochrome display. The tabular presentation was selected because there are important attributes associated with the entities displayed on the map display, which would severely clutter the map if included there.

USER: "Where is the Dresden airbase?"

CUBRICON:

Color Graphics Display:

- o Map of Inner Fulda Gap Region with added area that includes the Dresden airbase (Figure 2).

Item	Disposition	Latitude	Longitude	Name	Mobility
Fighter base	enemy	51 1000	11 5000	Dresden	---
Fighter base	enemy	51 1000	11 5000	Netzerby	---
Fighter base	enemy	51 1000	11 5000	Grossenhain	---
Fighter base	enemy	50 9800	12 5100	Strehla	---
Fighter base	enemy	51 4000	11 4600	Altenstein	---
Fighter base	enemy	50 9100	10 4600	Erfurt	---
Fighter base	enemy	51 1000	12 5400	Brandis	---
SA-2	enemy	51 0800	12 4600	---	low
SA-2	enemy	51 1000	13 1100	---	low
SA-2	enemy	51 1400	12 7100	---	low
SA-2	enemy	50 9800	11 5900	---	low
SA-2	enemy	50 9300	10 9300	---	low
SA-3	enemy	51 1000	13 0000	---	high
SA-3	enemy	50 9100	12 1400	---	high
SA-3	enemy	51 4000	11 1100	---	high
SA-3	enemy	50 9100	10 1400	---	high
SA-4	enemy	51 4700	12 9300	---	low
SA-4	enemy	51 4000	11 9100	---	low
SA-4	enemy	51 5100	13 1500	---	high
SA-4	enemy	51 0400	13 1500	---	high
SA-4	enemy	51 5100	13 1500	---	high
SA-4	enemy	51 1400	13 1200	---	high
SA-4	enemy	51 2100	13 4000	---	high
steel plant	friendly	51 2100	13 4000	Schütz steel	---
steel plant	friendly	51 4200	9 1400	Zerogora steel	---
munitions factory	friendly	50 9400	13 1500	Schwarz munitions	---
munitions factory	friendly	49 9900	10 1100	Erz munitions	---

p> Display the FG region
 p> Where is the Dresden airbase?
 p>

Run 11 Jul 8:43:02 pw11c1 CL SHEPS: User Input USER

- o Main roads, major cities, waterways, and national boundaries (as before but across the whole map, old and new areas) (Figure 2).
- o Icons representing entities within the map area displayed that are above a pre-set threshold in importance are superimposed on the map (same as before but across the whole map, old and new areas) (Figure 2).
- o An airbase icon labelled "Dresden Airbase". This airbase icon is displayed as initially blinking and then highlighted.

Monochrome Graphics Display:

- o Table of relevant entity attributes. Same table as before, but expanded to include new entities added to the map (Figure 2). The Dresden Airbase table entry (row) is highlighted.

DISCUSSION:

The information requested is location. The Dresden airbase has an icon representation and a longitude, latitude associated with it in the CUBRICON knowledge base. The preferred modality for presentation is therefore graphical.

In composing a new map on which to display the Dresden airbase, the system has some choices. These include: open a window on the color-graphics display showing the area around the Dresden airbase, replace the old map on the CRT with a new area around the airbase, or compose a new map including both the old map and the region around the Dresden airbase.

An important principle that CUBRICON tries to follow is to preserve the context of the human-computer dialogue. Since the user task has not changed and there is already a map displayed on the color graphics display, the system expands the displayed area to include the Dresden airbase. CUBRICON selects this option because it provides the requested information while it preserves the display context. When the Dresden airbase is added to the map, the system also blinks it as the system's way of "pointing" to the object under discussion.

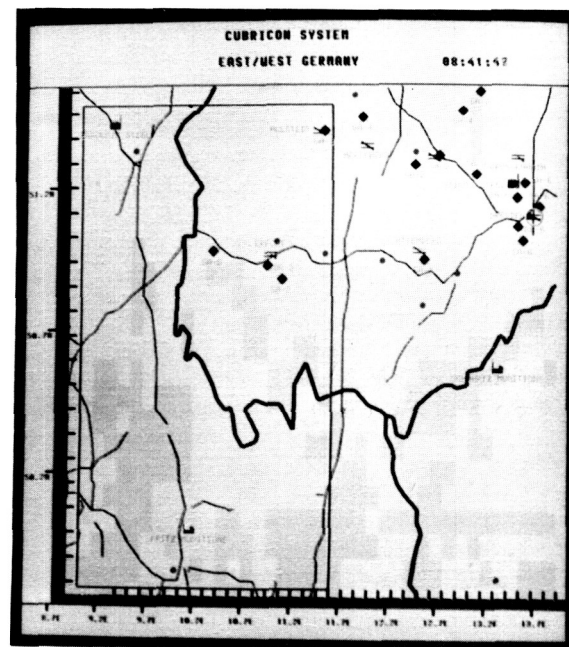


Figure 2. Map and Table Maintaining Context and Consistency

Another important guideline to which the CUBRICON system adheres is to maintain consistency throughout a display so as to prevent the user from making false inferences about what is or is not located within the region. In the case of our map display, this means that there should be consistency in the types of objects shown across the entire map. If SAMs are displayed in the old region, then they should be displayed in the newly added map area. Similarly for other types of objects. If this is not done, then the user might incorrectly infer that there were no SAMs in the new area. Guided by the consistency principle, the system also modifies the tabular presentation that is on the monochrome display. The new displays are shown in Figure 2.

USER: "What is the mobility of this SAM (accompanied by a point gesture to a SAM with the mouse device)?"

CUBRICON:

Spoken and Written Natural Language:

- o "The mobility of this SA-3 is high." The phrase "this SA-3" is accompanied by blinking the particular icon as the system's means of pointing to it.

DISCUSSION:

CUBRICON selects natural language as the modality as per the criteria presented in Section 3.5. This example also demonstrates CUBRICON's ability to generate multi-media output that is highly integrated. Specifically, in this example, the spoken natural language definite reference "this SA-3" is coordinated with a blinking reference on the graphics display.

USER: "What are the mobilities of these <point1><point2><point3><point4><point5>?" The five pointing gestures indicate five different SAM icons on the color map display.

CUBRICON:

- o Color-Graphics Display: The referenced SAMs are highlighted.
- o Monochrome Display: A window containing a table displaying the mobilities of the indicated SAMs is added.

DISCUSSION:

The user's question is almost the same in these last two examples. CUBRICON responds in coordinated NL and pointing when the response is a single proposition, as in the previous example. However, CUBRICON selects a table modality with associated highlighting on the color-graphics display for this last response. This selection is made since the values of a common attribute (mobility) of several entities must be expressed. This situation matches the selection criteria for the table modality as presented in Section 3.5.

6. CURRENT STATUS AND FUTURE DIRECTION

The work discussed in this paper has been implemented on the hardware suite described in Section 2. The knowledge sources discussed in Section 3 have been implemented and the functionality

described in this paper has been realized in the current CUBRICON prototype. The examples presented in Sections 4 and 5 are working examples.

The CUBRICON team is continuing its research and development of the concepts and methodology essential to intelligent multi-media human-computer interface systems. This includes continued research and development of the knowledge sources such as the user/task model and discourse model, the automated process of determining the appropriate media/modalities for any given information items or clusters, appropriate modification of the displays including placement of information when needed, the multi-modal composition process, and the role of speech output in a multi-modal interface system.

7. SUMMARY

Modern information processing and decision-aiding systems are complex and require a full range of communication media to facilitate interaction and provide the increased bandwidth for information transfer with the human user. The human-computer interface must be able to use and manage the media and modalities in an intelligent manner. The Intelligent Multi-Media Interface Project is devoted to the development of interface technology that integrates speech, natural language text, graphics, and pointing gestures for human-computer dialogues. The objective of the project is to develop interface technology that uses the media/modalities intelligently in a flexible, context-sensitive, and highly integrated manner modelled after the manner in which humans converse in simultaneous coordinated multiple modalities. As part of the project, a knowledge-based interface system, called CUBRICON is being developed as a research prototype. Several knowledge sources are essential to intelligent use of multiple modalities and are included in our CUBRICON system: a knowledge base of application-specific and related interface knowledge, a definition of the multi-media language in the form of a lexicon and grammar, a discourse model, a user/task model, knowledge of information characteristics and the appropriate corresponding modality for expressing the information, and human factors guidelines for enhancing human understanding and reducing cognitive workload. CUBRICON accepts dual-media input consisting of natural language and simultaneous coordinated pointing gestures. The CUBRICON methodology handles the synergistic mutual disambiguation of simultaneous natural language and pointing as well as inconsistent NL/pointing expressions and expressions that have an apparent null referent. CUBRICON's output composition process includes selection of appropriate modalities and media, determination of whether resources are available, subsequent modification of resources or modification of the information to be expressed (if necessary), modification of selected output media/modalities (if necessary), and composition of the output. Examples were presented to illustrate some of the key functionality of the CUBRICON system.

8. ACKNOWLEDGEMENTS

The other members of the CUBRC team are: D. Funke, S. Glanowski, M. Summers, and S.C. Shapiro.

The MAP Display System was developed by Lt. Mike Hilton (RADC), who graciously allowed us to use it.

9. REFERENCES

1. Arens, Y., Miller, L., Sondheimer, N.K., "Presentation Planning Using an Integrated Knowledge Base," *Proceedings of the Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Lockheed AI Center, Monterey, CA, 1988, pp. 93-107.
2. Carberry, S., "First International Workshop on User Modeling," *AI Magazine*, Vol.8, No.3, 1987, pp. 71-74.
3. Grosz, B. J., "Discourse Analysis," in *Understanding Spoken Language*, D. Walker (ed.), Elsevier North-Holland, New York, 1978, pp. 229-345.
4. Grosz, B.J. & Sidner, C.L., "Discourse Structure and the Proper Treatment of Interruptions," *Proc. of IJCAI*, 1985, pp. 832-839.
5. Grosz, B.J., "The Representation and Use of Focus in a System for Understanding Dialogs," in *Readings in Natural Language Processing*, B.J. Grosz, K.S. Jones, B.L. Webber (eds.), Morgan Kaufmann Publishers, 1986, pp. 353-362.
6. Hilton, M.L., *Design and Implementation of the MAP Display System*, RADC Report, 1987.
7. Kobsa, A., Allgayer, J., Reddig, C., Reithinger, N., Schmauks, D., Harbusch, K., Wahlster, W., "Combining Deictic Gestures and Natural Language for Referent Identification," *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, FR Germany, 1986.
8. Neal, J.G., Dobes, Z., Bettinger, K.E., & Byoun, J.S., "Multi-Media References in Human-Computer Dialogue," *Proceedings of the National Conference of the American Association for Artificial Intelligence*, ST. Paul, MN, 1988, (forthcoming).
9. Neal, J.G. & Shapiro, S.C., "Intelligent Multi-Media Interface Technology," *Proceedings of the Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Lockheed AI Center, Monterey, CA, 1988, pp. 69-91.
10. Neches, R. & Kaczmarek, T., *AAAI-86 Workshop on Intelligence in Interfaces*, USC/Information Sciences Institute, August, 1986.
11. Shapiro, S.C., "The SNePS Semantic Network Processing System". In *Associative Networks - The Representation and Use of Knowledge by Computers*, N. Findler (ed.), Academic Press, New York, 1979, pp. 179-203.
12. Shapiro, S.C., "Generalized Augmented Transition Network Grammars for Generation from Semantic Networks," *AJCL*, Vol. 8, No. 1, 1982, pp. 12-25.
13. Shapiro, S.C. & Rapaport, W., "SNePS Considered as a Fully Intensional Propositional Semantic Network," *Proceedings of AAAI-86*, 1986, pp. 278-283; in *Knowledge Representation*, G. McCalla & N. Cercone (eds.), Springer-Verlag Pub.
14. Sidner, C.L., "Focusing in the Comprehension of Definite Anaphora," in *Computational Models of Discourse*, M. Brady & R.C. Berwick (eds.), The MIT Press, 1983, pp. 267-330.
15. Smith, S.L. & Mosier, J.N., *Guidelines for Designing User Interface Software*, MITRE Corporation Technical Report No. 10090, ESD-TR-86-278, 1986.
16. Sullivan, J.W. & Tyler, S.W. (eds.), *Proceedings of the Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Lockheed AI Center, Monterey, CA, 1988.

Knowledge-Based Graphical Interfaces for Presenting Technical Information

Steven Feiner

Department of Computer Science
Columbia University
New York, NY 10027

feiner@cs.columbia.edu

Abstract

Designing effective presentations of technical information is extremely difficult and time-consuming. Moreover, the combination of increasing task complexity and declining job skills makes the need for high-quality technical presentations especially urgent. We believe that this need can ultimately be met through the development of knowledge-based graphical interfaces that can design and present technical information. Since much material is most naturally communicated through pictures, our work has stressed the importance of well-designed graphics, concentrating on generating pictures and laying out displays containing them.

We describe APEX, a testbed picture generation system that creates sequences of pictures that depict the performance of simple actions in a world of 3D objects. Our system supports rules for determining automatically the objects to be shown in a picture, the style and level of detail with which they should be rendered, the method by which the action itself should be indicated, and the picture's camera specification. We then describe work on GRIDS, an experimental display layout system that addresses some of the problems in designing displays containing these pictures, determining the position and size of the material to be presented.

Keywords: knowledge-based graphics, user interface design, graphical layout, design grids

1. Introduction

Technical information design and delivery systems based on paper and microfilm are gradually being replaced by computer-based systems. Conventional approaches to designing the user interfaces to these new systems and the information that they manage typically rely on handcrafted dialogues and parameterized displays. As a consequence, they are expensive and time-consuming to produce, much like the older systems that they replace.

The author's current work is supported in part by the Defense Advanced Research Projects Agency under Contract N00039-84-C-0165, the New York State Center for Advanced Technology under Contract NYSSTF-CAT(87)-5, and an equipment grant from the Hewlett-Packard Company. The IGD and APEX systems were supported in part by the Office of Naval Research under Contract N00014-78-C-0396 and the National Science Foundation under Grant INT-7302268-A03.

One way to improve the interface design process is to use graphical editors, rather than programming, to specify the appearance and interaction capabilities of the user interface. This concept was developed in systems such as [HANA80; FEIN82; WONG82; BUXT83; GREE85; OLSE85] and has since been borrowed and popularized by the recently introduced HyperCard [GOOD87].

1.1. Editor-Based Design

Editor-based systems have shown some dramatic results in allowing users, both programmers and nonprogrammers, to design certain kinds of interfaces in less time than it would take using conventional methods. In addition to increasing design throughput, editor-based systems can also increase design quality by encouraging successive refinement. If some part of the initial design is deemed inadequate it may be relatively easy to modify it.

IGD (Interactive Graphical Documents) [FEIN82; FEIN88a] is an early example of an experimental editor-based interface design system. Its users create interactive graphical hypermedia presentations that are designed and presented on a high-resolution color monitor. Pictures and typeset text are created with one editor and incorporated into a presentation with another. A display from a sonar maintenance and repair manual created with IGD is shown in Figure 1. This display, as well as the rest of the manual, was designed by a team of authors and illustrators who used the system's graphical editors, rather than a programming language. The same graphical editor that is used to specify the visual appearance of the display is also used to determine the display's interactive capabilities. For example, the designer can graphically select objects and make them into buttons that when touched cause actions to be performed, such as jumping to a new display. The editor provides a display of the manual's structure that allows users to both build and view interconnections between all of the displays.

1.2. Problems with Editor-Based Design

Our experience, and that of users of commercial systems like Hypercard, have shown that systems of this sort are powerful tools for trying out interface ideas. Unfortunately, there are

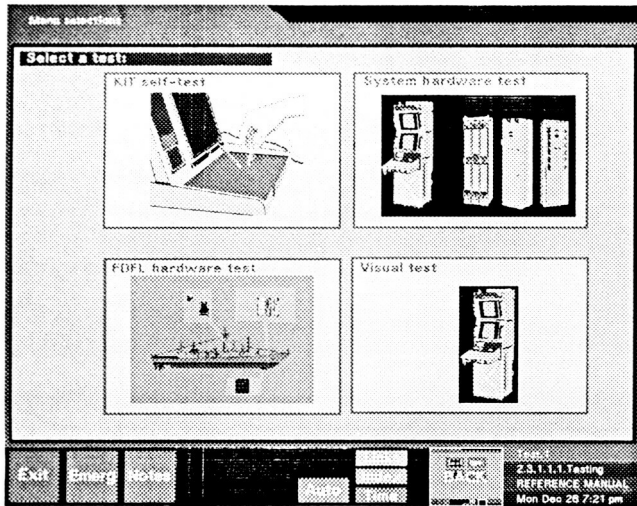


Figure 1. An interactive display from a manual created with IGD.

several difficulties with using them for creating large-scale systems. The first is caused by the need for customization. Editor-based interface design systems require that the interface designer anticipate all users, information, and situations that will be handled. One attempt to meet this need relies on the careful crafting of different sets of responses and presentations for a small number of equivalence classes. For example, users might be divided into novices, intermediates, and experts, and parts of the presentation designed differently to accommodate each. Rough equivalence classes like these, however, do not adequately reflect the large, heterogeneous, and changing user population that a large-scale system may have.

A second problem is raised by the need for immediacy. Timely, on-the-fly presentations of unanticipated information are essential for C^3 , as well as for technical documentation, if these applications are to cope with unanticipated information and presentation needs. If a human designer is involved in adapting the system when such situations arise, the turnaround time for presenting information will be unacceptable.

We believe that the ultimate solution to these problems is the automated generation of both the form and content of the information delivered. In the research reviewed in this paper we have concentrated on the design of explanatory pictures and the layout of displays containing these pictures. Thus, although our work has been in a maintenance and repair domain, our emphasis has not been on determining what actions to perform (i.e., on automated troubleshooting), but rather on explaining to the viewer how to perform those actions. Related work on automating the generation of pictures

and displays includes [ZDYB81; FRIE84; MACK86; AREN88; NEAL88].

2. Automating Picture Generation

Our work in picture generation has resulted in the creation of a testbed system, described here, that creates sequences of pictures that depict the performance of simple actions in a world of 3D objects [FEIN85]. APEX (Automated Pictorial EXplanations) is designed to mediate between an AI problem solver and conventional graphics software, as shown in Figure 2. The problem solver has expertise about a maintenance and repair domain and can develop a plan for fixing a piece of broken equipment that involves rigid body transformations (translation and rotation) of its parts. The graphics software is capable of drawing pictures of scenes whose contents and camera specification are explicitly described to it.

APEX takes as input the same information about the objects in the world and what the user knows about them that is provided to the problem solver, as well as the plan for the actions to be performed on the objects that is determined by the problem solver. APEX produces as output the specifications for a set of pictures to be generated by the graphics software that can be used to explain these actions to the repair person.

Rather than simply passing the entire existing environment of objects along with a camera specification to the graphics software, APEX instead builds a new environment. The goal is to create an environment whose picture will be more effective at communicating desired information than a picture of the original environment. Although the new environment is based on the original environment, objects may be selectively included, excluded, or even created from scratch. For example, an object may be left out if it is not related to the task to be illustrated, resulting in a simpler, less cluttered picture.

APEX supports rules for determining automatically the objects to be added to the new environment, the style and level of detail with which they should be rendered, the method by which the action itself should be indicated, and the picture's camera specification. We refer to this process of building a new environment to create a more effective picture as *depiction* [FEIN87].

2.1. Depicting Objects

APEX starts with an initially empty environment and adds the following kinds of objects, which it selects by processing the objects in the original environment.

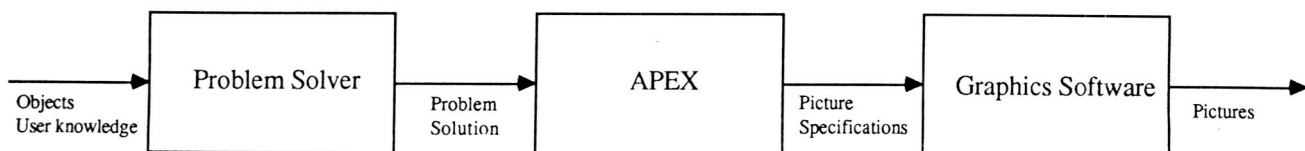


Figure 2. APEX converts the problem solver's plan into specifications of pictures to be drawn by the graphics software.

Frame objects. Each of APEX's pictures is designed to show a particular action being performed. The empty picture crystallizes around a small set of objects that directly participate in the action. We call these the picture's *frame objects* since they are specified by the action frame [MINS75] being depicted.

Context objects. Next, APEX adds objects that will provide context for those objects that are already included. The objects in APEX's world form a hierarchy. Context objects are selected by traveling up the hierarchy starting with each object that was originally included in the picture. Objects encountered are added to the picture up until the first object with which the user is already familiar.

Landmark objects. Although the context objects are helpful, they are often not sufficient to help locate the frame objects and may themselves be difficult to recognize. Therefore, APEX's picture-making strategy searches for *landmark* objects that could serve as a reference in locating those objects that have been included in the picture thus far. It does this by examining the objects that are near the important objects and selecting those that have significantly different appearance as determined by their shape, size, or the material from which they are made.

Similar objects. APEX searches the environment for nearby objects that are similar in appearance to those already included. These are added to the picture to help eliminate the chance that the viewer will confuse them with the objects included so far.

Supplementary objects. Additional objects are added in order to assure that the picture looks correct. For example, objects that physically support objects that are already in the picture are added so that the supported objects don't seem to be floating unsupported.

Meta-objects. In order to show the action being performed in a picture and to help distinguish the objects affected, APEX creates additional objects that are added to the picture. These *meta-objects* are arrows that are used to show translational and rotational motion. At the same time, the position of the arrow also indicates the object being moved.

2.2. Depicting Properties

When APEX adds an object to the picture it also determines several properties: camera specification, rendering style, and level of detail.

Camera specification. The picture's camera specification is modified for each added object to determine how much of the object should be visible. APEX's rules force frame objects, context objects, landmarks, similar objects, and meta-objects to be entirely visible. Supplementary objects, on the other hand, either cause no change in the camera specifications or may cause relatively small changes to enable some portion of them to be visible.

Rendering style. APEX selects the rendering style used for each object. Currently only two styles are employed. The

first, the "regular" rendering style, causes objects to be depicted with their actual material properties. This is used for frame objects, context objects, and meta-objects. A "subdued" rendering style is assigned to all other objects that are added to the picture to indicate that they are less important. APEX currently realizes a subdued style by blending the object's material properties (which determine its rendered color) with the properties of its parent.

Level of detail. APEX determines the level of detail to be used in rendering an object. Only enough detail is used to disambiguate an object from others that are similar in appearance to it. Much work on APEX was devoted to developing a method for determining automatically physical approximations of objects that could be used to depict them at different levels of detail.

Figure 3 shows a picture designed by APEX to show the viewer that they are to open the drawer of the center equipment cabinet by pulling on its middle handle. The cabinet itself was included to serve as context for the drawer that is part of it. The small cabinet on the wall was added as a landmark and the floor was included as a supplementary (supporting) object. The large cabinets on both sides were added because of their similarity to the center cabinet, while the top and bottom handles were included because of their similarity to the middle handle. Just enough detail was used in depicting objects to disambiguate them from those objects that were decided to be similar to them. A meta-object arrow shows that the drawer is to be pulled out.

Depiction is a general concept whose application is not limited to making pictures of 3D environments. In other work, we have applied it to the creation of editable graphical histories for user interfaces. We have designed a graphical editor that displays a pictorial "comic strip" history of the user's interactions [KURL88]. Each "panel" of the history is created using rules similar to those used by APEX to determine automatically the objects to include and how they should be rendered. Unlike APEX, multiple actions are compacted into a single panel when appropriate. The user can interact graphically with the history to review their session and to undo, modify, and redo past actions.

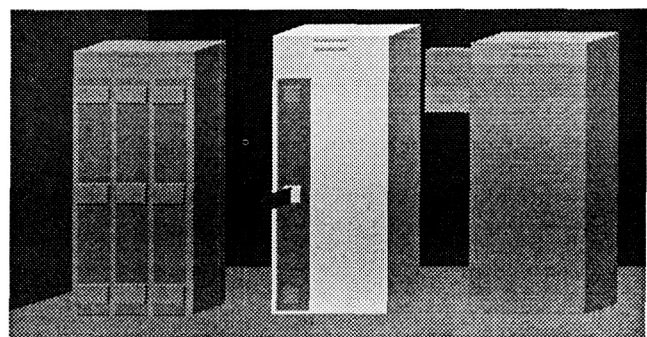


Figure 3. A picture designed by APEX to show how to open the center cabinet's drawer using its middle handle.

3. Automating Display Layout

In order to create a coherent and effective presentation, not only must pictures (and text) be created, but they must be combined together on the display in a process known as *display layout*. Our current work in display layout treats some of the problems in determining the position and size of material that is to be presented to the user. We have developed a testbed system called GRIDS (G^Raphical I^Nterface D^Esign S^Ystem), which lays out displays containing pictures and text, determining the size and position of the parts from which they are composed [FEIN88c].

GRIDS takes as input information about the objects to be displayed, the user, and the display hardware. It uses this to determine a layout that will be applied to each screenful of objects. Its approach is based on the idea of grid-based layout developed by graphic designers [HURL78; MULL81]. A *design grid*, consisting of proportionally-spaced horizontal and vertical lines, is imposed on the space to be laid out. The lines describe a set of rectangular grid *fields*. The fields are separated vertically and horizontally by equal-sized spaces and the array of fields is surrounded on all four sides by margins. Objects are sized and positioned on the grid in such a way that they are aligned with the grid lines. Thus each object is positioned in a part of the grid that is an integral number of fields in height and width.

3.1. Designing a Layout

Our system generates a grid and determines how objects will be placed using it. Its approach is briefly reviewed here and described in more detail in [FEIN88c]. First a grid is created, based on input information about the material to be laid out, the display, and the user. For example, the user's distance from the display constrains the sizes of the fonts and pictures that can be used, while the size and aspect ratio of the physical display constrain both the size and relative position of the objects to be laid out. These in turn help determine the size of the grid's fields, margins, and inter-field spaces.

Next, the grid that the system produces is used in conjunction with input information about the objects to be laid out to generate a *prototype display layout*. An important part of this input information is a grammar that describes the kinds of objects that will be included in the actual displays. The actual objects that will be presented in a particular display are instances of the general classes of objects that are the grammar's terminals. The system currently supports pictures, body text blocks, and headings. These objects are further specialized by designating limits on their expected size and content. The grammar also specifies grouping relationships among these objects. For example, the grammar may specify that displays can contain pictures that are each related to a block of text that serves as its caption. Finally, the prototype display layout is used to determine how to lay out input instances of the objects described by the display grammar to form the actual displays.

The GRIDS testbed is implemented in OPS5 [FORG81] and generates output in PostScript [ADOB85]. Figure 4 shows an example of a layout designed by GRIDS, with and without the

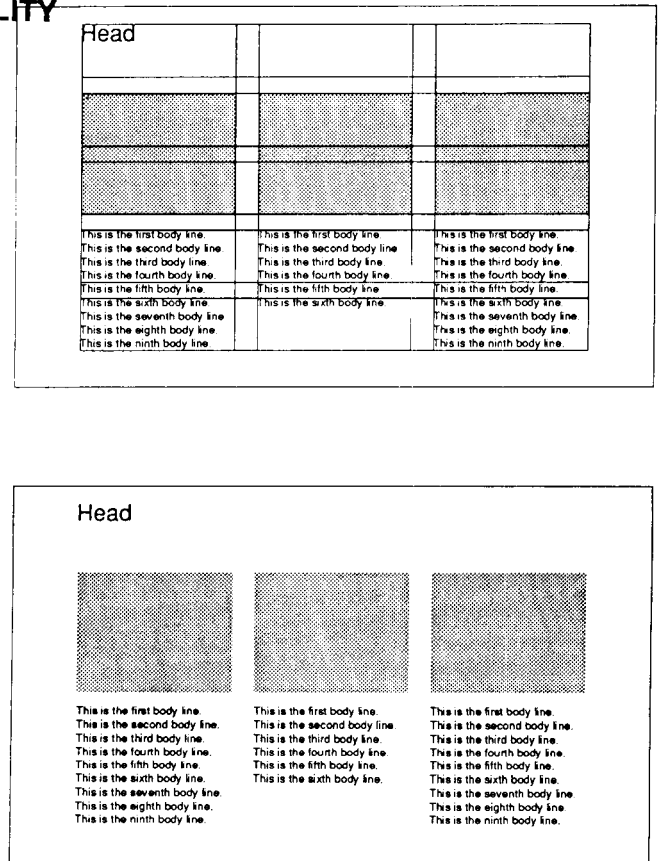


Figure 4. A display layout designed by GRIDS, with and without the grid used to generate it.

grid that was used to generate it. The outermost rectangles indicate the boundaries of the physical display. (The grid does not actually appear in the layout presented to the user.) GRIDS currently does not lay out actual pictures and text, but rather works with the approximations shown in the figure: rectangular shaded areas for pictures and numbered lines for text.

By generating a grid first and using it to produce multiple layouts, we gain one of the important advantages of grid-based design: consistency [MULL81]. Each display to be laid out is not optimized as an individual design problem, but bears a visual relationship to the other displays. Not only do we gain efficiency in not having to redesign each display afresh, but the use of a common layout format visually enforces the relationship between the displays. The limited selection of sizes and positions used within a single display also helps establish intra-display consistency.

4. Conclusions

We have described work in two aspects of automatically generating presentations of technical information. APEX creates sequences of explanatory pictures, while GRIDS designs and lays out displays containing separately created pictures and text. An important underlying theme of both

systems is that to ensure a consistent presentation, a common set of design rules should be provided or generated first, and then used to create individual pictures and displays.

The projects described here are partial, testbed implementations of a general conceptual architecture for generating both layout and information content automatically [FEIN88b]. Much work remains to be done to eventually develop robust, knowledge-based design systems that can produce timely, high-quality technical presentations that are customized to the needs of particular users.

References

- [ADOB85] Adobe Systems Inc. *PostScript Language Reference Manual*. MA: Addison-Wesley, 1985.
- [AREN88] Arens, Y., Miller, L., and Sondheimer, N. "Presentation Planning Using an Integrated Knowledge Base." *Proc. ACM/SIGCHI Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Monterey, CA, Mar 29-Apr 1, 1988, 93-107.
- [BUXT83] Buxton, B. "Toward a Comprehensive User Interface Management System." *Computer Graphics*, 17:3, July 1983, 35-42.
- [FEIN82] Feiner, S., Nagy, S., and van Dam, A. "An Experimental System for Creating and Presenting Interactive Graphical Documents." *ACM Trans. on Graphics*, 1:1 January 1982, 59-77.
- [FEIN85] Feiner, S. "APEX: An Experiment in the Automated Creation of Pictorial Explanations." *IEEE Computer Graphics and Applications*, 5:11, November 1985, 29-38.
- [FEIN87] Feiner, S. A framework for automated picture generation. Columbia Univ. Dept. of Comp. Sci. Tech. Rep. CUCS-277-87, 1987.
- [FEIN88a] Feiner, S. "Seeing the Forest for the Trees: Hierarchical Display of Hypertext Structure." *Proc. COIS88 (ACM-SIGOIS/IEEE Comp. Soc. Conf. on Office Info. Sys.)*, Palo Alto, March 23-25, 1988, 205-212.
- [FEIN88b] Feiner, S. "An Architecture for Knowledge-Based Graphical Interfaces." *Proc. ACM/SIGCHI Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Monterey, CA, Mar 29-Apr 1, 1988, 129-140.
- [FEIN88c] Feiner, S. "A Grid-Based Approach to Automating Display Layout." *Proc. Graphics Interface '88*, Edmonton, June 6-10, 1988 (Morgan Kaufmann, Palo Alto, 1988), 192-197.
- [FORG81] Forge, C. *OPS5 User's Manual*. Computer Science Technical Report CMU-CS-81-135, Carnegie-Mellon University, July 1981.
- [FRIE84] Friedell, M. "Automatic Synthesis of Graphical Object Descriptions." *Computer Graphics*, 18:3, July 1984, 53-62.
- [GOOD87] Goodman, D. *The Complete HyperCard Handbook*, NY: Bantam Books, 1987.
- [GREE85] Green, M. "The University of Alberta User Interface Management System." *Computer Graphics*, 19:3, July 1985, 205-213.
- [HANA80] Hanau, P. and Lenorovitz, D. "Prototyping and Simulation Tools for User/Computer Dialogue Design." *Computer Graphics*, 14:3, July 1980, 271-278.
- [HURL78] Hurlburt, A. *The Grid*. NY: Van Nostrand Reinhold Co., 1978.
- [KURL88] Kurlander, D. and Feiner, S. "Editable Graphical Histories." To appear in *Proc. 1988 IEEE Comp. Soc. Workshop on Visual Languages*, October 10-12, 1988, Pittsburgh, PA.
- [MACK86] Mackinlay, J. "Automating the Design of Graphical Presentations of Relational Information." *ACM Trans. on Graphics*, 5:2, April 1986, 110-141.
- [MINS75] Minsky, M. "A Framework for Representing Knowledge." In P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, 1975, 211-277.
- [MULL81] Müller-Brockmann, J. *Grid Systems in Graphic Design*. Niederteufen, Switzerland: Verlag Arthur Niggli, 1981.
- [NEAL88] Neal, J. and Shapiro, S. "Intelligent Multi-Media Interface Technology." *Proc. ACM/SIGCHI Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes*, Monterey, CA, Mar 29-Apr 1, 1988, 69-91.
- [OLSE85] Olsen Jr., D., Dempsey, E., and Rogge, R. "Input/Output Linkage in a User Interface Management System." *Computer Graphics*, 19:3, July 1983, 191-197.
- [WONG82] Wong, P., and Reid, E. "FLAIR - User Interface Dialog Design Tool." *Computer Graphics*, 16:3, July 1982, 87-98.
- [ZDYB81] Zdybel, F., Greenfield, N., Yonke, M., and Gibbons, J. "An Information Presentation System." *Proc. IJCAI 81*, Vancouver, August 24-28, 1981, 978-984.

**AN INTELLIGENT INTERFACE FOR SATELLITE OPERATIONS:
YOUR ORBIT DETERMINATION ASSISTANT (YODA)**

Anne Schur
GE/RCA Advanced Technology Laboratories
Route 38, Moorestown Corporate Center
Moorestown, New Jersey 08057

ABSTRACT

An intelligent interface is often characterized by the ability to adapt evaluation criteria as the environment and user goals change. Some factors that impact these adaptations are redefinition of task goals and, hence, user requirements; time criticality; and system status. To implement adaptations affected by these factors, a new set of capabilities must be incorporated into the human-computer interface design. These capabilities include: 1) dynamic update and removal of control states based on user inputs, 2) generation and removal of logical dependencies as change occurs, 3) uniform and smooth interfacing to numerous processes, databases, and expert systems, and 4) unobtrusive on-line assistance to users of varied skill levels. This paper discusses how these concepts were applied and incorporated into a human-computer interface using artificial intelligence techniques to create a prototype expert system, YODA (Your Orbit Determination Assistant). YODA is a "smart" interface that supports in real time orbit analysts who must determine the location of a satellite during the station acquisition phase of a mission. The paper also describes the integration of four knowledge sources required to support the orbit determination assistant: orbital mechanics, spacecraft specifications, characteristics of the mission support software, and orbit analyst experience. This initial effort is continuing with expansion of YODA's capabilities, including evaluation of results of the orbit determination task.

INTRODUCTION

Current satellite mission support tasks at GE utilize off-line mission support software known as SOCS (Spacecraft Orbit Control System). SOCS stores information about the satellite and contains the algorithms used for nine major analysis tasks; e.g., orbit determination and ephemeris propagation. These algorithms operate on parameter values supplied by the orbit analyst (O/A). Entries are made via a terminal, with no input prompts provided. It is the responsibility of the O/A to check all inputs and consult a set of manuals for input requirements, including appropriate format and values. Once all

values have been entered, the O/A issues a command telling SOCS to receive the entered values and perform the selected analyses. When an analysis is complete, SOCS returns an output via hardcopy. The O/A must interpret the results to determine their validity.

These practices are time consuming and require the O/A to have experience in orbital mechanics, a sound knowledge of the spacecraft characteristics, and knowledge of both the SOCS software and data entry procedures. Compounding these difficulties is the trend for future satellite mission support tasks to serve an increased number of satellites of increased complexity, despite the counter trend toward satellite autonomy. As the total mission support workload increases, personnel will be in shorter supply, and therefore, often lacking in experience. It will be difficult for these personnel to respond to and perform critical tasks in a timely manner using current practices. To alleviate this situation, GE took steps to simplify ground support operations by creating an intelligent human-computer interface. The objective of this effort was to reduce both workload and required experience level needed to perform orbit determination.

An intelligent interface is often characterized by the ability to adapt evaluation criteria as the environment and user goals change. Satellite orbit determination is a reoccurring mission support task in which the criteria used to locate satellites changes with environmental changes; e.g., variations in spacecraft characteristics, orbit and mission phase. The adaptable interface developed for this task was named YODA* (Your Orbit Determination Assistant). YODA is an expert system prototype that assists orbit analysts who must determine the location of a satellite under severe time constraints, who have little real-time satellite operations experience, or who may perform the orbit determination task infrequently.

*YODA was implemented on a Texas Instrument Explorer using ART (Automation Reasoning Tool, developed by Inference). Common LISP was used to reformat the input orbit determination values into a form that was readable by the SOCS software.)

YODA has the ability to check all inputs, adapt its evaluation criteria, dynamically update and remove control states based on user inputs, and provide unobtrusive on-line assistance to users of various skill levels. The following discussion describes how these capabilities were incorporated into the human-computer interface.

USER OPERATIONAL REQUIREMENTS

For YODA to be a success it was critical for it to be accepted by the user community. Therefore, priority was given to meeting user operational requirements. Table I lists these requirements, together with the features implemented in YODA to meet the specified needs.

Collectively, these features impart intelligence to the user interface. Knowledge incorporated into the human-computer interface includes the ability to sense/make inferences about: appropriate default values, advice needed, value constraints for sanity checks, and the relationships between items as a function of the situation.

TABLE I. USER REQUIREMENTS AND ASSOCIATED YODA FEATURES

REQUIREMENT	YODA FEATURE
Perform only orbit determination tasks	<ul style="list-style-type: none"> Minimizes required user knowledge of computer systems, SOCS and data input procedures Provides default values, and limit and sanity check criteria
Easily accommodate user change of mind	<ul style="list-style-type: none"> Adapts with mind change
Serve multiple users <ul style="list-style-type: none"> Reduce need to seek expert advice 	<ul style="list-style-type: none"> Provides relevant help easily
Allocate control to user versus system	<ul style="list-style-type: none"> Enables multipath access Performs any task in any order
Provide feedback <ul style="list-style-type: none"> what to do result of input 	<ul style="list-style-type: none"> Assesses value of user input and procedure choice based on current situation Meaningful input prompts, error messages, and advice based on current situation Advice when required
Provide knowledge, "drowned in information but starved for knowledge"	<ul style="list-style-type: none"> Cognitive model of tasks Maintains relationships between information items
System expandable to on/off-line training	<ul style="list-style-type: none"> Track where the user is in the task (future)
Grow system to accommodate more spacecraft and SOCS modules	<ul style="list-style-type: none"> Standardized operations Underlying code readable, repeatable, updatable, without requiring close support from AI expert

APPROACH

Knowledge Acquisition

Knowledge acquisition was performed using subject matter experts from two technical domains: SOC software and orbit determination, because current practices require the O/A to have knowledge of both these domains. For example, communicating a value for a particular element requires the O/A to input SOCS values indicating the particular record, the specific memory location, and the associated target element value. Table II shows the domain knowledge needed to make an entry. The information actually input to SOCS is highlighted. Figure 1 shows a completed orbit determination input with the examples provided in Table II highlighted.

To meet user requirements, a detailed task analysis was included as part of the knowledge acquisition process. This approach enabled an operational concept to be formulated that specifically met the needs of the O/A in the context of all mission support phases (see Figure 2). Additionally, the O/A would be provided with a means to create and maintain a cognitive model of the orbit determination task and would no longer be required to know SOCS software.

TABLE II. EXAMPLES OF ORBIT ANALYST KNOWLEDGE REQUIRED TO INPUT ORBIT DETERMINATION INFORMATION

DOMAIN KNOWLEDGE					
Orbit Determination		SOCS Software			
Input Type	Item	Perm-Up Data Type	SOCS Blocks	Memory Location	Value
Bounds	Element 1	PE	SREALS	354	1000000
Coordinate System	Keplerian				6
Spacecraft	SPACENET				A

Knowledge Acquisition Results

The task analysis revealed the orbit determination function to be comprised of six distinct operational tasks. Two of the tasks were perceived by the O/A to contain optional subtasks; one subtask, for example, is adding special modifications which enable effects, such as solar pressure, drag, time bias, and range bias to be considered during the orbit determination computation performed by the SOCS software. Also, the task analysis provided the framework upon which to associate all information items and establish the relationships between identified items. Each item upon which the analyst would perform some action required that the following kinds of information be associated with it:

```

➤ SEMIDIAMETERDEMO-1ASODO      15      0
TAB105TAB104TAB103TAB102TAB101TAB100TAB099TAB098TAB097TAB096TAB095TAB094TAB093
TAB092TAB091
PE,SREALS,371,50
PE,SREALS,370,50
PE,SREALS,369,50
PE,SREALS,368,1000000
PE,SREALS,367,1000000
PE,SREALS,366,1000000
PE,SREALS,359,100.0
PE,SREALS,358,100.0
PE,SREALS,357,100.0
PE,SREALS,356,123
PE,SREALS,355,0.05
➤ PE,SREALS,354,1000000
PE,SREALS,49,0.9
0,2,122388,233445
➤ 4021,6,2345565543,0.72988194
0,456,567,458
0,3,0,0
'''
'''
'''
'''
'''
,0,2,2
end

```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 1. Currently Used Orbit Determination Input Display

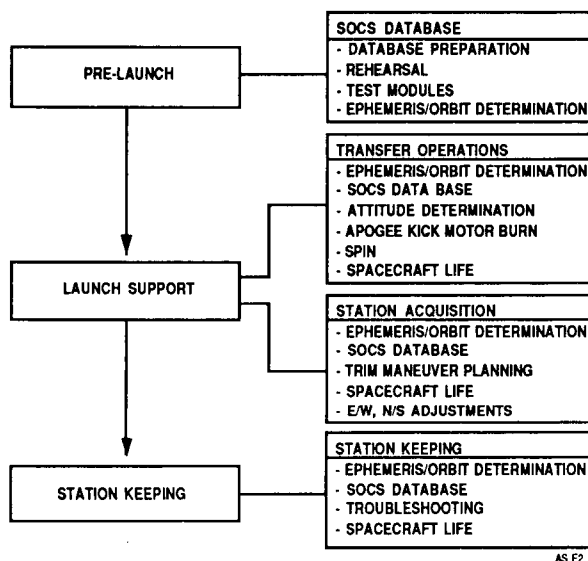


Figure 2. Mission Support Phases for Satellite Operations

- criteria to access information
- criteria to remove or exchange information
- input messages
- error messages
- general help (task level; e.g., initial guess, using YODA)
- specific help (item level; e.g., coordinate system selection)
- evaluation criteria in respect to the set-up environment
- where to transfer information, if applicable
- SOCS values.

Figure 3 is a simplified diagram of these relationships. Tasks are indicated as rectangles, with the information associated with each task and the relationships between each information item depicted hierarchically. Thus, if an information item occurring above another is removed, then the information items below it will also be removed and/or exchanged for appropriate new information.

SOFTWARE DESCRIPTION

Overall System

Figure 4 shows a system function flow. The O/A enters values, which YODA checks for validity. When all inputs have been made, the analyst indicates that they may be sent to SOCS for orbit determination computation. This action calls a routine which converts all the values specified by the analyst into a form that can be read by the SOCS software (Figure 1). The SOCS software then computes the location of the satellite and provides the results in hardcopy. It is up to the analyst to validate the orbit determination results. (An expert system to perform this latter task is now being developed. The outputs of this expert system will be fed into YODA in the form of recommendations to improve the orbit determination result.)

Software Architecture

The interface architecture is composed of objects and rules. The objects are represented in two separate knowledge-bases: a task knowledge-base and a value knowledge-base. The task knowledge-base is composed of those items (constants) which will always be part of the task despite user initiated (rule activated) changes to those objects: i.e., their values. This knowl-

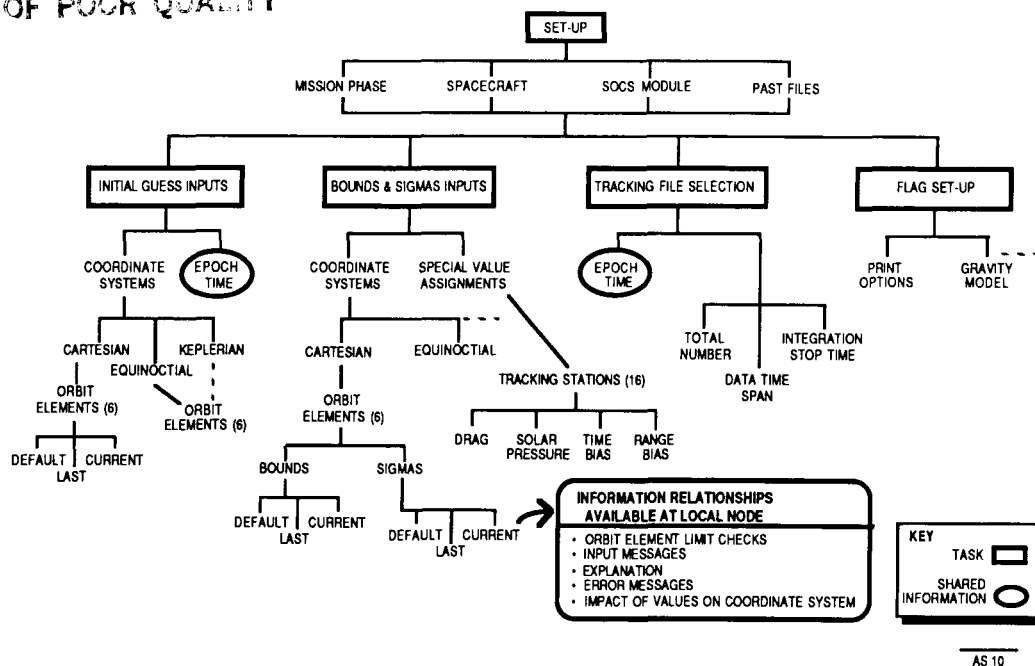


Figure 3. Simplified Diagram Showing Information Relationships Between Individual Items and Orbit Determination Tasks

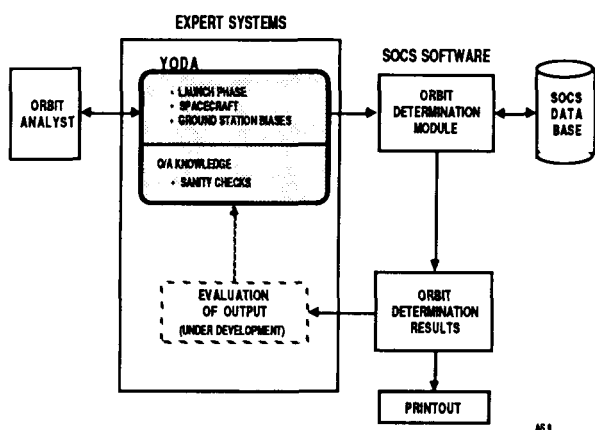


Figure 4. System Level Function Flow

edge-base plays a critical role in achieving the criterion of providing a cognitive model of the orbit determination process to the analyst. Figure 5 shows an example from the analyst's perspective. The display/window is represented as an object and corresponds to a specific orbit determination task. The icons/objects within the window represent the items needed by the analyst to perform the task. Figure 3 represents the way in which the task knowledge-base is structured. The value knowledge-base contains the values which the objects in the constant knowledge-base can adopt, and the evaluation criteria used to assess the validity of any values input made by the analyst (see Figure 6). Specific input messages, error messages, and recommendations are also housed in this knowledge-base. A third knowledge-base contains the general help facilities. Table III compares, in summary

TABLE III. COMPARED OBJECT CHARACTERISTICS OF THE TASK AND VALUE KNOWLEDGE-BASES

TASK KNOWLEDGE-BASE OBJECTS	VALUE KNOWLEDGE-BASE OBJECTS
• Directly manipulatable by analyst, empty input slots	• Analyst unable to manipulate these objects directly
• Associated SOCS function values	—
• Control state identifiers	—
• Links to associated defaults, evaluation criteria, and help information housed within other knowledge-base	• Corresponding links for default and evaluation criteria
—	• Default values, evaluation criteria, input and error messages, all of which correspond to appropriate mission phase, spacecraft, coordinate system, etc.

form, the differences between the objects of the task knowledge-base and the value knowledge-base.

The rules embody the knowledge of the interface. Each rule refers to the state of some relevant knowledge-base. In accordance with a change of state initiated by the O/A the rule can adapt the interface to the appropriate context. For example, using an extreme case, if an analyst changes his/her mind about which spacecraft upon which to perform an orbit determination, YODA can update and/or remove appropriate control

ORIGINAL PAGE IS
OF POOR QUALITY

BDSG
Vehicle ID: A
BOUND AND SIGMA VALUES
Socs Mode: SODO

ELEMENT CORRECTION
 Select Correction:
 1. Cartesian
 2. Equinoctial
 3. Keplerian

 Keplerian

SCALE FACTOR
 Select factor:
 1. Increasing
 2. Decreasing

 Decreasing 0.35

VALUE SOURCE	DEFAULTS	LAST RUNDECK Cartesian	LAST UNITS	CURRENT
BOUNDS				
a (n)	1000000	1000000	Rx (n)	1000000
e (-)	0.05	1000000	Ry (n)	0.05
i (deg)	0.5	100.0	Rz (n)	0.5
o (deg)	3.0	100.0	Vx (n/sec)	0.07
u (deg)	3.0	100.0	Vy (n/sec)	3.0
n (deg)	5.0	100.0	Vz (n/sec)	5.0
SIGMAS				
a (n)	100000	1000000	Rx (n)	100000
e (-)	0.0e-4	1000000	Ry (n)	0.0e-4
i (deg)	0.2	1000000	Rz (n)	0.2
o (deg)	1.0	50	Vx (n/sec)	-----
u (deg)	1.0	50	Vy (n/sec)	-----
n (deg)	1.0	50	Vz (n/sec)	-----

HELP TOP MENU
NEXT DISPLAY Display ID

DEFINITION FOR SIGMAS

SIGMAS

Sigmas are those standard deviations for each element of the initial guess element set. A large sigma indicates little confidence in the value of the initial guess element, and likewise a small sigma indicates high confidence in the value.

Figure 5. Graphical Input Display Used to Support Bound and Sigma Value Entry Task

```

(DEFSCHEMA KEPLERIAN-BOUND-DEFAULTS-FOR SPACENET
  "default values for Keplerian bounds of SPACENET"
  (vehicle-id A)
  (use correction-type)
  (class bounds)
  (coordinate-system keplerian)
  (element (el1 1000000))
  (element (el2 1.05))
  (element (el3 0.5))
  (element (el4 3.0))
  (element (el5 3.0))
  (element (el6 5.0)))
  
```

Figure 6. Example of an Object in the Knowledge-Base

states and generate and/or remove logical dependencies as the change occurs. All element values (defaults, last, current) associated with the 'old' vehicle will be removed from the fact base. Default and last file values associated with the 'new' spacecraft in respect to the already specified coordinate system, and the constraints used for last file selection (mission phase, SOCS software module) will be displayed. The analyst will be expected to input new current values. At a local level, changing a coordinate system will result in the O/A seeing an exchange of default values and the removal of current values. Evaluation criteria and user advice will also be appropriately removed and/or exchanged, but in a manner that is transparent to the analyst. Figure 7 shows the information that is presented to the analyst when asking about the Cartesian coordinate system. To obtain this information the analyst pointed to the word "Cartesian". The rules also maintain the appropriate values required to execute

SOCS functions. This bookkeeping task is also transparent to the user.

User Interaction with YODA

The dialogue between the analyst and YODA consists of direct manipulation. This allows the analyst to control the sequence of events and therefore have the freedom to take any action in any order. An expert could fill in orbit determination values working within a single display (the summary display). A less experienced analyst can walk through each task in a top down sequence, while a more experienced analyst can perform the tasks nonsequentially. At all times the analyst is provided with commands that are relevant to the task. To enable the value knowledge-base to be easily updated, schemata were written in English (Figure 6). To change the values, the current values have to be deleted and new ones put in their place. The

person performing this update would not have to modify the rule base.

SUMMARY

YODA is a knowledge entry program developed to allow orbit analysts to efficiently use an external data source. This is accomplished by enabling orbit determination evaluation criteria to be correctly specified by an O/A so that orbit determination computations can be executed by a software system. The O/A works within a visual programming environment. Graphical forms allow specification of knowledge using a "fill in the blanks" approach. For example, orbit element values are represented in a schema language as simple icons. Whenever one of these icons is selected with a mouse, a special environment is entered, providing inferential knowledge needed to evaluate or provide advice about the parameter value in respect to the situation. The O/A is at all times in control of the input process. This is achieved by interactive displays that support the analyst's cognitive model of the task in a natural idiom. Feedback to the O/A is immediate and is presented in a manner which conveys the actions taken by the system on the inputs supplied by the analyst.

The developed prototype was successfully demonstrated. It is now being ported into a delivery system to enable YODA's use in GE's mission control room during daily operations and satellite launches.

FUTURE

The current system is limited to creating inputs for entry into the SOCS software. The output from the orbit determination computation is evaluated by the analyst. An effort is currently under way to automate this task, using an expert system that will inform the O/A of the results of the output evaluation and recommend changes that could be made on the initial inputs. It is also planned to expand this system to an on-line job-performance aid, where the effect of the orbit parameter values input by the O/A can be seen dynamically and in comparison to both the known defaults and last parameter value sets for the specified mission phase, spacecraft, and coordinate system.

REFERENCES

ASC Off-line Software User's Guide, RCA Corporation, Astro-Electronics Division, Princeton, NJ, Document Number: UG-OV8-2606915, 1986.

Bayman, P., and Mayer, R.E., "Instructional Manipulation of Users' Mental Model for Electronic Calculators," *International Journal of Man-Machine Studies*, 20, 189-199, 1984.

Chandrasekaran, B., "Expert Systems: Matching Techniques to Tasks," *Artificial Intelligence in Business*, Editor Reitman, W. Norwood NJ: Ablex, 1984.

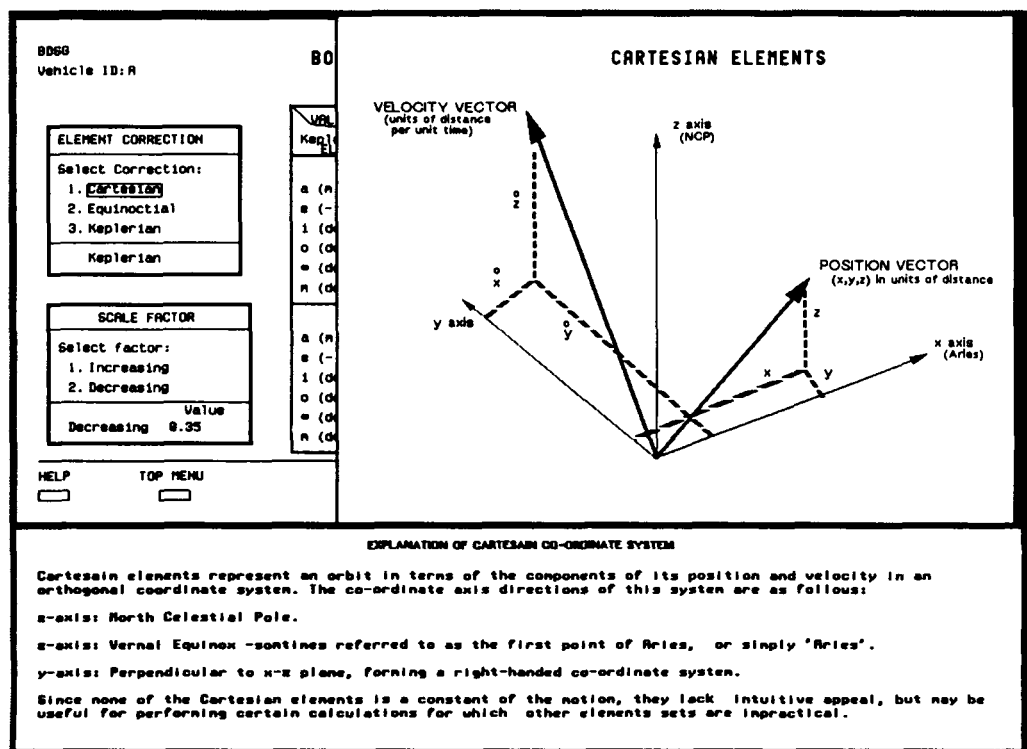


Figure 7. Explanation Display Provided When Requested by Analyst Pointing to "Cartesian"

Charniak, E.C., "Toward a Model of Children's Story Comprehension," MIT AI Laboratory, TR-266, Cambridge, MA, 1972.

Hendler, J.A., (Ed), *Expert Systems: The User Interface*, Norwood NJ:Ablex, 1987.

Norman, D.A., *Cognitive Engineering. In User Centered Design*, Editors: Norman, D.A., and Draper, S.W., Hillsdale NJ:Erlbaum, 31-65, 1986.

Rasmussen, J., *Information Processing and Human Machine Interaction*, Elsevier Science Publishing Co, North-Holland Series, 1986.

Shneiderman, B., "Direct Manipulation: A Step Beyond Programming Languages," *Computer*, 16 (89), 57-69, 1983.

SOCS Database Manual, RCA Corporation, Astro-Electronics Division, Princeton, NJ, Document Number: IS-AV7-2606915, 1987.

Swigger, K. M., "An Intelligent Tutoring System for Generation of Ground Tracks," **Proceedings of the American Association for Artificial Intelligence**, 72-76, 1987.

AN INTELLIGENT TUTORING SYSTEM FOR SPACE SHUTTLE DIAGNOSIS

William B. Johnson
 Jeffrey E. Norton
 Phillip C. Duncan
 Search Technology, Inc.
 4725 Peachtree Corners Circle
 Norcross, Georgia 30092

ABSTRACT

Intelligent Tutoring Systems (ITSs) transcend conventional computer-based instruction. An ITS is capable of monitoring and understanding student performance thereby providing feedback, explanation, and remediation. This is accomplished by including models of the student, the instructor, and the expert technician or operator in the domain of interest. The space shuttle fuel cell is the technical domain for the project described below.

One system, Microcomputer Intelligence for Technical Training (MITT), demonstrates that ITSs can be developed and delivered, with a reasonable amount of effort and in a short period of time, on a microcomputer. The MITT system capitalizes on the diagnostic training approach called Framework for Aiding the Understanding of Logical Troubleshooting (FAULT) (Johnson, 1987). The system's embedded procedural expert was developed with NASA's CLIPS expert system shell (Culbert, 1987).

MITT was conceived and sponsored by the Air Force Human Resources Laboratory, Brooks Air Force Base, Texas. The research, development, and evaluation of MITT was completed with cooperation from NASA at the L.B. Johnson Space Center, Houston, Texas.

INTRODUCTION

Intelligent Tutoring Systems (ITSs) are instructional systems that deliver training in a manner comparable to that of a human tutor. ITSs deliver instruction, interact with students, and structure subsequent instruction based on student performance. To provide such instruction, ITSs must contain an understanding of a specific domain, a means to model student understanding of that domain, and a component containing pedagogical guidelines for providing feedback and remediation. These components must surround an instructional environment. The instructional environment must have a reasonable

interface to the student user. The components of a generic ITS are shown in Figure 1 (Johnson, in press). There are many recent publications that offer thorough definitions of ITSs and describe existing systems (Psozka, Massey, & Mutter, 1988; Polson & Richardson, 1988; and Wenger, 1987).

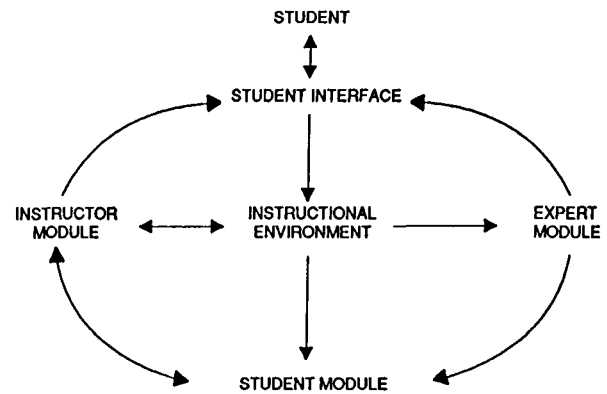


Figure 1. Components of an Intelligent Tutoring System

This paper describes a six-month research effort in which mature computer-based instruction (CBI) for diagnostic training was integrated with the NASA expert system shell, C Language Integrated Production System (CLIPS), to create a fully operational ITS on a microcomputer. The system is called Microcomputer Intelligence for Technical Training (MITT).

Although ITSs have been around for nearly twenty years, they have not emerged as common components in training departments. Instead, they have remained as research topics in industry, government, and university laboratories (Johnson, in press). The purpose of the MITT project was to prove that ITSs can be developed in a reasonable amount of time for a reasonable price. Also, the MITT developers wanted to show that ITS technology is sufficiently mature to contribute to an operational training environment.

COMPUTER-BASED INSTRUCTION

To develop an instructional environment for MITT, Search Technology, Inc. capitalized on its more than 10 years of research experience with computer-based simulation for diagnostic training. That research (Hunt & Rouse, 1981; Johnson, 1981, 1987; Johnson & Rouse, 1981; Rouse and Hunt, 1984) was characterized by an evolving set of computer simulations and extensive experimental and real-world evaluations. This R&D was conducted in diverse domains, such as automotive mechanics, aviation mechanics, communication/electronics, and nuclear safety systems. Search Technology's CBI research began as an attempt to understand how humans gather and process information in problem-solving situations. This led to research into the effects of training on problem-solving behavior. From this research, a variety of training concepts were developed and evaluated in laboratory and field tests. The diagnostic training simulations that emerged from this research were Troubleshooting by Application of Structural Knowledge (TASK) and Framework for Aiding the Understanding of Logical Troubleshooting (FAULT). Both simulations are described by Rouse and Hunt (1984).

FAULT, proven in a variety of instructional domains, is at the heart of the MITT system. It uses a hard copy

functional flow diagram along with an on-line display for student options, test results, and feedback. FAULT is a simulation that permits the user to engage in the same information processing that would take place during real equipment troubleshooting. This includes actions such as checking instruments, obtaining symptomatic reports from an operator, forming hypothesis, selecting tests, and identifying parts for replacement. FAULT simulations developed prior to MITT included a limited degree of intelligence that provided student advice and feedback (Johnson, Norton, Duncan, & Hunt, 1988).

THE MITT SYSTEM

The MITT system consists of five parts shown in Figure 1: the instructional environment, the student interface, the expert module, the student module, and the instructor module. This section describes each of the modules and how they communicate to form the Intelligent Tutoring System.

The Instructional Environment

MITT's instructional environment is the FAULT simulation. Perhaps the greatest strength of the FAULT simulation is the simplicity of the system's representation. Figure 2 shows MITT's functional flow representation of the space shuttle fuel cell. Each part

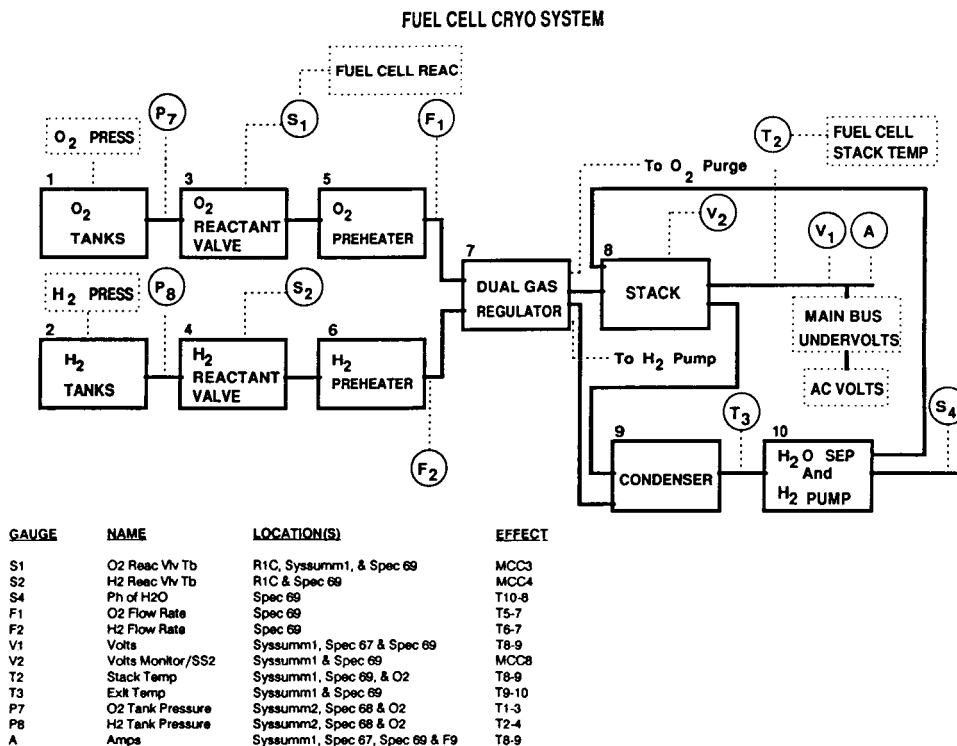


Figure 2. MITT Fuel Cell Functional Flow Diagram

in the system is represented as a node in a network. MITT's simple, single functional relationship helps the student create a mental model of the technical system.

The Student Interface

Although each module of an ITS is critically important, it is the user interface that ultimately delivers the instruction to the user. The interface must allow easy interaction with the simulation. Also, the outputs of the expert, student, and instructor modules must be completely integrated with the student interface in order to be effective and unobtrusive.

A mouse is the primary user input device, although a keyboard may also be used at any time. The cursor, controlled by either the mouse or the keyboard, can be moved within each menu to select student diagnostic options. On certain screens, extensive data are available as shown in Figure 3. For the simulation to know precisely what data the student is requesting, it is necessary for the user to click on the XXX area (see column 3 in Figure 3) to obtain the information.

2011/ /069		FUEL CELL										4 /02/11:48:21 000/00:00:00	
		1	2	3									
VOLTS		30.5	29.7	XXX	H2O RLF LINE	T		90					
AMPS		208	208	XXX	NOZT	A		210					
						B		210					
FLOW	D2	4.4	4.5	XXX	HTR SW								
H2		0.5	0.8	XXX	PURGE LN	O2	T	51					
REAC	O2	OP	OP	XXX	H2	T1		62					
H2		OP	OP	XXX		T2		39					
					H2O LINE PH			XXX					
STACK	T	+203	+204	XXX									
EXIT	T	150	150	XXX									
COOL	T	72	70	XXX									
P		60	60	XXX									
PUMP													
H2 PUMP		0.4	0.4	XXX			1	2	3				
READY	RDY	RDY	RDY	XXX	PH								
H2O LN	T	134	133	134	V SS1	17	11	14	XXX				
VLV	T	72	70	70	SS2	10	13	13	XXX				
EC	A				SS3	19	17	17					
HEATER	B												
F1 HELP		F7 BACK UP ONE					F9 BACK TO START						

Figure 3. CRT Display Requiring Specific Information Request Areas

The Expert Module

MITT's expert module represents the system in two ways. It has both a procedural and a functional expert. The procedural expert (PE) maps symptoms to suspect systems and components with collections of if-then rules. These symptoms include information such as annunciator lights and instrument indications. The functional expert (FE) uses one or more connectivity matrices to represent various functional relationships among the parts of the system.

The procedural expert

The procedural expert is a series of rule-based statements. It contains information in the form of advice regarding specific NASA troubleshooting

procedures for the space shuttle's fuel cell. At any stage during diagnosis of a malfunction, the student can request procedural advice. The PE gives advice according to the order in which gauges, controls, or panels have been seen by the student. Advice is given only when it is requested from the MITT menu by the student. This procedural advice suggests the next step to take in the diagnosis of a malfunction. The student may follow the advice that is given or choose from other menu options.

PE development required an analysis of existing NASA troubleshooting procedures. These procedures were broken down into steps or actions that are likely to be used by the student. Each action corresponds to the reading of simulated gauges, controls, or panels available in the shuttle. Search Technology's analysis of the MITT project had to ensure that the student had some means to accomplish a given NASA procedure within the constraints of the simulation.

Each statement of advice is linked to a logical test. For example, if the orbiter's primary annunciator panel (F7) has been seen and the front gauge panel (F9) or cathode-ray tube (CRT) display System Summary 1 has been seen, then an advice statement describes what has been seen, what conclusions can be drawn, and what actions are appropriate. All of the advice statements for the six malfunctions are incorporated into approximately 60 rules. Additional procedural intelligence can be incorporated by merely adding to the established rule base.

The functional expert

The functional expert (FE) is based on the functional connectivity among the system components. For example, if part B depends on part A and part A has failed, then part B will be adversely affected. Also, part B will adversely affect parts which depend on it. This functional understanding permits the expert to calculate how failures propagate through the system by following the functional topography.

The student has the option to interact with this expert by requesting FE advice while working on a problem. The FE provides assistance based on the functional structure of the system and the student's previous actions. The FE also communicates with the student module to monitor changes in the feasible set of failures based on each student action.

Expert integration

The experts must have the means to communicate with each other. During MITT's development, this was accomplished by creating an equivalency table between gauge readings and topographic tests. For example, a reading of the O2/H2 flow on a gauge may

be the same as a test between two parts of the functional flow diagram. As far as MITT was concerned, they are the same. All of the tests were translated to gauge readings to match the format of the PE. In addition, the gauge readings were translated to functional flow tests for the FE.

The Student Module

The student module creates a model of the student by tallying the student's actions throughout the simulation. The student model is updated by the FAULT simulation and the expert module. The data from the student model is used by the instructor module to determine when advice is appropriate.

MITT has a student model that is current for each problem. The model includes tests the student has taken and results of these tests. The model keeps track of the number of times the student uses an option from the primary simulation display as well as the number of accesses to each orbiter gauge, annunciator, or CRT. The model also keeps track of the type of errors noted by the functional expert. The student module provides feedback upon completion of each problem as shown in Figure 4.

Congratulations! You have corrected the problem.		
Number of students completing this problem:	78	
Number of students who quit before solving:	2	
Number of students who received a time out:	6	
	<u>Your</u>	<u>Average</u>
Number of minutes for diagnosis:	7	8.4
Number of errors made:	2	1.6
Number of displays accessed:	14	17.3
Number of times procedural advice used:	1	1.8
Number of times functional advice used:	2	1.3

Figure 4. MITT Feedback at Problem Completion

The Instructor Module

The instructor module is a rule-based routine that pinpoints certain student errors and intervenes as they occur. The types of errors detected include student actions that result from a student misunderstanding of either the MITT system or of simple troubleshooting procedures. The advantage to this approach is that the instructor module provides generic advice that promises to be effective for current and future simulations.

The instructor module was designed to help guide the student to the most appropriate segment of the MITT system. Instead of redundantly explaining something, it suggests where the student should look for more information. In this sense, the instructor module actually works more like a reference librarian than a

teacher. The advantage to this approach is that only those who need reference material use it.

The entire MITT system is designed to deliver instruction in such a way that even novices can easily use the system. The system includes help for using the MITT simulation and contains technical information and diagnostic advice on the fuel cell's domain. MITT is student driven with complete learner control. Only under exceptional circumstances does the instructor module intervene and redirect the student from exploring his own paths.

Hardware

The hardware system used for the development and delivery of the MITT ITS was an IBM-AT (or an IBM-compatible system). The system requires 640 Kb of random access memory (RAM) and a hard disk. Presently, an Color Graphics Adapter (CGA) card and color monitor are needed. A mouse is recommended, but optional.

The rationale for using IBM-ATs is straightforward. First, the equipment is affordable and more likely to be found in research laboratories and (more importantly) training installations. When a group decides to use MITT, they will not have to purchase expensive, dedicated AI workstations. Second, IBM-ATs have all the necessary speed, storage, hardware and software support, and other capabilities to deliver the required level of intelligent tutoring. In addition, IBM-ATs offer readily available off-the-shelf peripherals for interface to video disk and other computers.

CLIPS

MITT is written in C. By capitalizing on CLIPS, MITT processes rules for the various modules without using LISP. Therefore during MITT's development, CLIPS was a convenient tool to use for several reasons. It can be embedded into existing C code, it has a built-in inference engine, the rules are simple to create, and CLIPS itself is easy to learn. In addition, CLIPS is highly portable across computer systems. CLIPS runs on IBM-ATs and IBM compatibles, and it is inexpensive (i.e., free for government use).

The disadvantage to using CLIPS, during MITT's development, was that Search Technology did not have a reliable, compiled version of CLIPS at the time MITT was completed. The uncompiled CLIPS version caused a delay in the system's response time to the student. The new, compiled version of CLIPS will increase the speed with which procedural advice is given, thereby improving user acceptance of MITT.

C-41

MITT EVALUATION

Evaluation can be divided into two stages: formative and summative evaluations. Formative evaluation takes place during software design and development. Summative evaluation refers to the software's value following development. For this short effort, the formative evaluation was most important.

Formative Evaluation

The primary goal of formative evaluation is to keep potential users informed as to ongoing development and the expected final product. Formative evaluation permits developers, subject matter experts, and prospective users to be constantly informed and able to make real-time changes in design. It also prevents the notion that "it is too late to change that now."

The majority of MITT's formative evaluation was accomplished by the ongoing interaction between the developers and personnel from NASA and AFHRL. During the program's development, Search Technology followed a proven evaluation plan for CBI development (Maddox & Johnson, 1986). The plan ensured that software evaluation was performed using a three-step process: measuring compatibility, understandability and effectiveness. Compatibility refers to the extent to which the user is able to see the computer displays and reach the pointing devices. Displays must be legible, and all colors must be easily discernible. Understandability is concerned with ITS output and required user input. Users must be able to understand what the system is telling them and what they must tell the system. In addition, the system input requirements must align with the student's prior knowledge and training. Effectiveness is similar to summative evaluation which is explained below.

Summative Evaluation

Summative evaluation takes place once the software is complete. It is a summary concerned with the effect of the training on student performance. MITT is not ready for a classic summative evaluation because only the first 6-month phase of the work is finished. However, the first phase did undergo two pseudo-summative evaluations that served as a way to define the steps needed for the next phase of MITT's development, tentatively called MITT II.

The first stage of the summative evaluation was conducted by Search Technology and NASA. This evaluation took place over a 2-day period at the Johnson Space Center. During that time, 17 NASA employees used MITT. These people included astronauts, flight controllers, CBI developers, AI

researchers, technical instructors, and a training manager. The goal of the evaluation was to obtain a preliminary assessment of user acceptance and to insure that the ITS was complete and technically correct. User acceptance was overwhelmingly positive.

The second stage of the summative evaluation was conducted by AFHRL and NASA, again at Johnson Space Center. For this evaluation, MITT was used by 15 flight controllers for approximately 3 hours each. Again, the MITT ITS simulation was well received by the students. Many users commented that they saw value in the problem-solving approach to training presented by MITT. The users were impressed with the fact that they could proceed at their own pace with their own problem-solving style. The students were also satisfied with the advice they received from the functional, procedural, and instruction experts.

The second evaluation pointed out areas of the MITT system that could be improved in subsequent versions of MITT. For example, it would be worthwhile to modify the on-line shuttle schematics to more closely resemble the functional flow diagrams. Some users commented that they would prefer higher fidelity displays of shuttle instrumentation. In addition, the current MITT problem knowledge base must be embellished to include a greater variety and number of problems.

Comments from NASA's instructional staff and training managers have been very positive. NASA is anxious to proceed with MITT II's development. Their first concern is to enhance the fuel cell knowledge base so that it can provide daily operational training to supplement existing classroom and simulator training. They also want the capability to build knowledge bases for additional space shuttle and space station subsystems. A goal of the MITT II program is to develop software tools and knowledge engineering techniques that will place an increasing amount of ITS development in the hands of training department personnel. These tools will help to decrease the costs of ITS development and further increase ITS availability.

CONCLUSION

MITT has clearly demonstrated that Intelligent Tutoring Systems can be developed in a reasonably short period of time at a reasonable cost. MITT has also shown that ITSs can be developed and delivered using off-the-shelf microcomputers. Most importantly, the MITT project has demonstrated that ITSs do not have to remain "laboratory rats", but instead can be viable components of operational training departments.

ACKNOWLEDGMENTS

The MITT project was sponsored by the Air Force Human Resources Laboratory at Brooks Air Force Base in Texas. The authors wish to acknowledge the cooperation, technical contributions, and support of Lt. Col. Hugh Burns, Capt. Kevin Klein, and Lt. Charles Capps. We also appreciate the cooperation of Ms. Barbara Pearson and Mr. Ken Swiatkowski from the NASA Johnson Space Center Operations Training Department. Finally, we would like to thank Dr. Ruston Hunt and Ms. Connie McFarland of Search Technology, Inc. for their technical input and support given during MITT's development.

References

- Culbert, C.J. (1987). *CLIPS Reference Manual* (JSC - 225522). Houston, TX: National Aeronautics and Space Administration Lyndon B. Johnson Space Center.
- Hunt, R.M., & Rouse, W.B. (1981). Problem solving skills of maintenance trainees in diagnosing faults in simulated power plants. *Human Factors* 23(3), 317-328.
- Johnson, W.B. (in press). Intelligent tutoring systems: If they are such good ideas, why aren't there more of them? *Proceedings of the Tenth Annual Interservice Industry Training Systems Conference*. Orlando, FL: November 1988.
- Johnson, W.B. (1988). Pragmatic considerations in development and implementation of intelligent tutoring systems. In M. Polson and R.J. Richardson (Eds.), *Foundations of intelligent tutoring systems*, (pp. 191-207) Hillsdale, NJ: Lawrence Erlbaum Associates.
- Johnson, W.B. (1987). Development and evaluation of simulation-oriented computer-based instruction for diagnostic training. In W.B. Rouse (Ed.), *Advances in man-machine systems research: Vol. 3*, (pp. 99-125) Greenwich, CT: JAI Press.
- Johnson, W.B. (1981). Computer simulations for fault diagnosis training: An empirical study of learning from simulation to live system performance. (Doctoral Dissertation, University of Illinois, 1980), *Dissertation Abstracts International*, 41(11). 4625-A. (University Microfilms No. 8108555).
- Johnson, W.B., Norton, J.E., Duncan, P.C., & Hunt, R.M. (1988). Development and demonstration of microcomputer intelligence for technical training (MITT): Phase 1 final report (AFHRL Tech. Rpt., in press). Brooks AFB, TX: Air Force Human Resources Laboratory.
- Maddox, M.E., & Johnson, W.B. (1986). "Can you see it? Can you understand it? Does it work? An evaluation plan for computer-based instruction. *Conference Proceedings: Advances in Human Factors in Nuclear Power Systems* (pp. 380-389). LaGrange, IL: American Nuclear Society.
- Polson, M.C., & Richardson, J.J. (1988). *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Psotka, J., Massey, R., & Mutter, S. (1988). *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rouse, W.B., & Hunt, R.M. (1984). Human problem-solving in fault diagnosis tasks. In W.B. Rouse (Ed.), *Advances in man-machine systems research: Vol. 1*. Greenwich, CT: JAI Press.
- Wenger, E. (1987). *Artificial intelligence and tutoring systems*. Los Altos, CA: Morgan Kaufmann.

Intelligent Tutoring Systems Research
in the
Training Systems Division: Space Applications

J. Wesley Regian, Ph.D.
Air Force Human Resources Laboratory
Training Systems Division

The CBT Continuum. Computer-Aided Instruction (CAI) is a mature technology used to teach students in a wide variety of domains. The introduction of Artificial Intelligence (AI) technology to the field of CAI has prompted research and development efforts in an area known as Intelligent Computer-Aided Instruction (ICAI). In some cases, ICAI has been touted as a revolutionary alternative to traditional CAI. "With the advent of powerful, inexpensive school computers, ICAI is emerging as a potential rival to CAI." (Dede & Swigger, 1987) In contrast to this, one may conceive of Computer-Based Training (CBT) systems as lying along a continuum which runs from CAI to ICAI. Although the key difference between the two is intelligence, there is no commonly accepted definition of what constitutes an intelligent instructional system (VanLehn, 1986).

For my purposes, I discriminate among CBT systems according to the degree to which the instruction they provide is individualized. My choice of this particular dimension is based on more of a desire for utilitarianism than for precision. A great deal of data from the traditional educational world indicates that one-on-one tutoring is superior to both mastery teaching and conventional teaching (Woolf, 1987; Bloom, 1984). Thus, an important way in which CBT systems differ is in the degree to which their behavior is modified by an inferred "model of the student's current understanding of the subject matter." (VanLehn, 1986) The CBT system that is less intelligent by

this definition, I conceive of as CAI. Similarly, the system that is more intelligent, I conceive of as ICAI. Often, ICAI systems are referred to as Intelligent Tutoring Systems (Sleeman & Brown, 1982). In this paper, I will refer to a single ICAI system as an ITS, and to multiple ICAI systems as ITSs.

With respect to individualization, it is important to note that virtually all traditional CAI systems are individualized in the sense that they are self-paced, and many are further individualized by virtue of branching routines which allow different students to receive different instruction. CAI systems with branching routines are, in fact, more individualized than those without branching routines. Thus, they are more intelligent by the current definition (although in a weak sense, as we shall see). Nevertheless, in branched CAI the instructional developer must explicitly encode the actions generated by all possible branches, and there is a finite number of possible paths through these branches. As one moves further away from the CAI to the ICAI end of the continuum, one begins to see a very different and more powerful approach to individualization. This more powerful approach is touched on by Wenger (1987) when he refers to explicit encoding of knowledge rather than encoding of decisions (pg. 4). An ITS (which term I reserve for systems which are very far toward the ICAI end of the continuum) utilizes a diverse set of knowledge bases and inference routines to "compose instructional interactions dynamically, making decisions by reference to the

ORIGINAL PAGE IS OF POOR QUALITY

knowledge with which they have been provided" (Wenger, 1987; pg. 5).

The ITS Anatomy. In an ITS, individualized instruction is an emergent property of several interacting components. ITSs often consist of four, sometimes five, distinct components. These are the expert module, the instructional module, the student model, the interface, and a device simulation when relevant.

The expert module is a programmed representation of expert knowledge in the target domain (that which is being taught). It is almost identical to what is commonly known as an expert system, except in this context it is often very articulate (able to generate some form of rationale for its actions) and capable of generating alternative solution paths (rather than a single 'best' path). The expert module brings domain knowledge to the ITS. In some useful sense, the system 'knows' how to perform the task which it is seeking to teach, and can demonstrate that knowledge.

The instructional module is a programmed representation of expert knowledge on pedagogy in the target domain. It is generally not articulate but is invariably capable of generating alternative instructional approaches based on the current knowledge level of the current student. While the expert module invariably derives from knowledge engineering with an expert practitioner in the target domain; the instructional module may derive from knowledge engineering with an expert instructor in the target domain (which may or may not be the same person as the expert practitioner), with a general training specialist, or both.

The student model differs from the expert and instructional modules in that it is a mere shell at the beginning of a tutoring session, whereas the latter two are robust and complete when the

development of the ITS is complete. At the beginning of a tutoring session the student model is merely a place to store specific kinds of information about students in particular formats that will be useful for the instructional module to access. The student model is dynamically updated during tutoring sessions to maintain current information about the student such as what the student knows, what the student does not know, and misconceptions the student may have. The student module brings situational awareness to the ITS. Thus, the system 'knows' who it is teaching to, and can make informed decisions about how to teach.

The interface provides the methods by which the student interacts with the ITS. The interface may include such output methods as computer generated graphics and text, recorded video images, or speech synthesizers; and such input devices as a mouse, keyboard, touchscreen, joystick, or voice recognition system. One important point about the interface is that it should be as simple as possible so that learning to use the ITS does not interfere with learning from the ITS.

Some of the ITSs developed at the Intelligent Systems Branch (e.g., MITT, MATIE), and many ITSs in general (e.g., STEAMER, IMTS/Bladefold, Sherlock) utilize an embedded computer simulation of an electrical or mechanical device, and thus provide device-specific instruction. The device simulations are used to teach operation or maintenance of a specific device in the context of an operating model of the device. Other ITSs teach a body of knowledge that is not specific to any particular device (e.g., SCHOLAR, LISP Tutor, Smithtown).

Knowledge Engineering. One of the bottlenecks in the development of an ITS is the process that has come to be known as knowledge engineering. The creation of any robust expert system (such as the

ORIGINAL PAGE IS OF POOR QUALITY

expert module of an ITS) requires a great deal of front-end work before a single line of code is written. The knowledge engineer must first discover how the expert performs the target task. What knowledge is required? How are subgoals defined and achieved? What inferences are made and from what data? For complex tasks, the process is arduous even with a very articulate expert. One of the hallmarks of expertise, however, is a reduced ability to separate and articulate the small steps of a complex cognitive operation (Anderson, 1983). Thus, the process of knowledge engineering tends to be a successive approximation leading slowly toward a complete model of the task. The Human Resources Laboratory (HRL), Naval Training Systems Center, and the Army Research Institute are jointly pursuing a program with the goal of providing tools to support the iterative process of knowledge engineering. The KA (Knowledge Acquisition) toolkit is a software package which allows the user to easily create a flowchart representation of a procedural task. The system then prompts the user to break the chart into smaller and smaller substeps, and to specify the inferences underlying decision points. The end result of a session with the KA toolkit is a running simulation of target task performance that is suitable to support training. The goal of KA is to automatically generate a running expert system from computer-aided knowledge engineering.

ITS Domains. Traditionally (if the term applies to a technology less than twenty years old), ITSs have focused on knowledge-rich tasks such as electronic troubleshooting, physics, economics, and medical diagnosis.

The **Orbital Mechanics (OM)** tutor, currently being developed at HRL, teaches students the device-independent body of knowledge known as orbital mechanics. For example, the Ground Tracks Curriculum Module

teaches the correspondence between orbital parameters and ground tracks of satellites. Ground tracks are displays which depict the changing relationship between the surface of the earth and the location of a satellite over time. Understanding this relationship is important for Satellite Operations Officers (2055 AFSC) who monitor and plan satellite missions. Because the OM curriculum is device-independent, the tutor provides appropriate instruction for any task requiring knowledge of orbital mechanics. It does not provide instruction on applying that knowledge in the context of a specific task using specific hardware.

The **Fuel Cell (MITT)** tutor, developed for HRL and NASA by Search Technologies, is an example of a device-specific tutor. MITT provides intelligent maintenance training for the fuel cells on-board the shuttle. This skill is important for Air Force Flight Controllers (20XX AFSC) and for space shuttle crew members. As a device-specific tutor, MITT is targeted for a specific point in the training curriculum of a specific group of students. That point lies midway between basic instruction and expensive simulation. For example, current NASA training for flight controllers involves a general systems course (Phase I), specialized Phase II instruction (e.g., Regency CAI and a workbook for fuel cell specialists), Single Systems Training (SST - individually tutored simulation time), and finally on-the-job training. The Phase II instruction is very inexpensive and very basic, whereas the SST component (also called SST malfunction class) is very expensive to provide at approximately \$600 per hour. SST utilizes a shuttle cockpit mockup and allows instructors to introduce various kinds of system failures into the simulation. MITT is an example of a low fidelity simulation-based ITS that is targeted to fill the gap between

ORIGINAL PAGE IS OF POOR QUALITY

Phase II and SST, so that students can learn a great deal about their task before moving on to the higher-fidelity, more expensive simulation. In this way they can make maximum use of their time on the more expensive, higher fidelity simulation.

Enabling Skills. In many cases, knowledge-rich domains involve components of expertise, sometimes called enabling skills, which can be characterized as high-performance or knowledge-lean components. For example, electronic troubleshooting involves schematic-tracing which is supported by the ability to immediately and accurately combine gate inputs to determine the output of a particular gate type as represented on the schematic. Similarly, expert performance in theoretical physics requires total facility with basic math and algebraic skills. Human instructors can recognize deficiencies in basic enabling skills (especially in one-on-one tutoring situations) and apply methods to correct these deficiencies.

ITSs as a rule are not sensitive to deficiencies in basic enabling skills, even though they are not difficult to identify. Moreover, computers are particularly well suited to providing the kind of drill-and-practice exercises that can correct the deficiencies. For example, in the Air Traffic Control (ATC) training regime for radar operators it is important to be able to visually estimate the angular heading of a radar blip within 5 degrees accuracy. This level of accuracy takes an average of 2000 training trials to achieve. Under normal training conditions, this many trials would require about 5.5 weeks of training time. In a computerized angle judgement module (Regan & Schneider, 1986), students perform a video-flash-card version of the task. In this form, students experience 2000 trials of the critical task in 3 hours.

In generating instruction for knowledge-rich domains, ITSs should be sensitive to the full range of performance determinants for the task, and have appropriate routines available for remediation. Furthermore, there may be a place for ITS technology even in relatively knowledge-lean domains, such as typing, air intercept control, and simple equipment operation. In these cases, knowledge engineering and subsequent knowledge representation for the target task would be relatively simple, since expert performance is defined more by skill than by knowledge. Conversely, knowledge engineering for the instructional module would require more emphasis.

In order to evaluate the utility of ITSs in knowledge-lean tasks, HRL is collaborating with the Southwest Research Institute in developing the Console Operations (COPS) tutor for NASA. COPS will be a prototype ITS to support the NASA Flight Control User Tools course, providing device-specific instruction for operators of the Propulsion Console. The first COPS module will teach the major components of the console, and console initialization procedures. Pedagogically, COPS focuses on the instructional principle of cognitive automaticity (Schneider, 1985). This principle describes the acquisition of cognitive skill with consistent practice. The goal of the first COPS module is to instill facility with the Propulsion Console so that the operator can then focus on the more knowledge-rich aspects of the task. Later COPS modules will teach console reconfigurations that are associated with key mission events (e.g., main engine cutoff, external tank separation, etc.). Currently in the design phase, this system represents a significant departure from the knowledge-rich domains which have traditionally been the focus of ITS development.

Conclusion. The infusion of AI technologies into CBT has the

**ORIGINAL PAGE IS
OF POOR QUALITY**

potential of producing CBT systems which provide individualized instruction rivaling the quality of one-on-one human tutoring. However, the incorporation of intelligent routines in CBT systems is expensive and should only occur when there is sufficient enhancement of instructional efficiency and effectiveness to offset the additional development cost. HRL is conducting research to help define the role of AI in training, to develop usable and maintainable systems for users, and to advance the spectrum of ITS applicability.

Woolf, B. P. (1987). A survey of intelligent tutoring systems. Proceedings of the Northeast Artificial Intelligence Consortium. Blue Mountain Lake, NY. June.

REFERENCES

- Anderson, J.R. (1983). The architecture of cognition. Cambridge, MA: Harvard University.
- Dede, C., & Swigger, K. (1987). The evolution of instructional design principles for intelligent computer-assisted instruction. Proceedings of the American Educational Research Association.
- Regian, J.W., & Schneider, W. (1986). Assessment procedures for predicting and optimizing skill acquisition. Paper presented at ETS conference on diagnostic monitoring, July.
- Schneider, W. (1985). Toward a model of attention and the development of automaticity. In M. I. Posner & O. S. Marin (Eds.), Attention and performance XI (pp. 475-492). Hillsdale, NJ: Erlbaum.
- Sleeman, D.H., & Brown, J.S. (Eds.) (1982). Intelligent tutoring systems. London: Academic Press.
- VanLehn, K. (1986). Student modeling in intelligent teaching systems. Proceedings of the Research Planning Forum for Intelligent Tutoring Systems. San Antonio, TX: Air Force Human Resources Laboratory.
- Wenger, E. (1987). Artificial intelligence and tutoring systems. Los Altos, CA: Morgan Kaufman.

ORIGINAL PAGE IS
OF POOR QUALITY

INTELLIGENT TUTORING IN THE SPACECRAFT
COMMAND/CONTROL ENVIRONMENT

Walter F. Truszkowski
Code 522.1
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

ABSTRACT

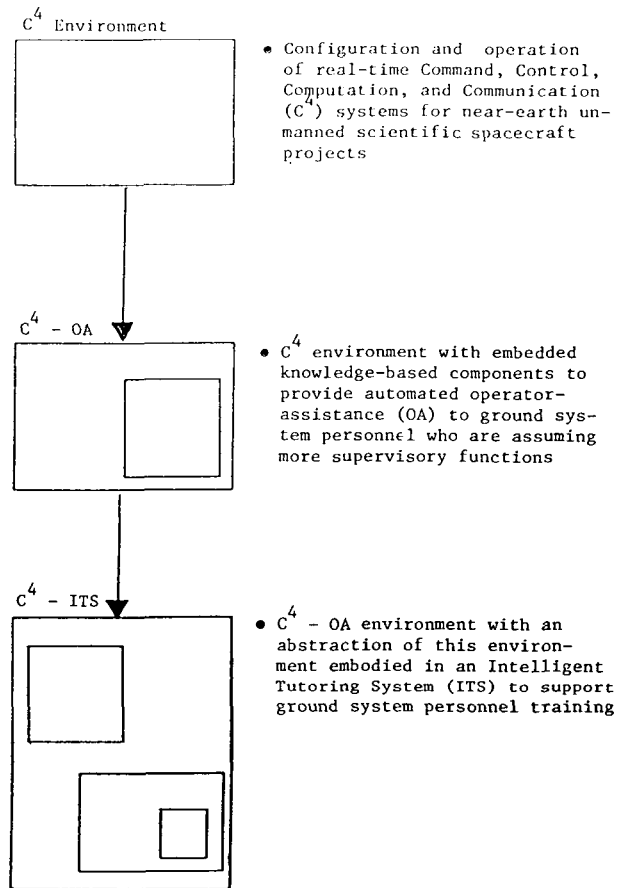
The spacecraft command/control environment is becoming increasingly complex. At this point in time as we are entering the era of Space Station and the era of more highly automated systems it is evident that the critical roles played by operations personnel in supervising the many required control center system components is becoming more cognitively demanding. In addition, the changing and emerging roles in the operations picture will have far-reaching effects on the achievement of mission objectives. Thus highly trained and competent operations personnel are mandatory for success.

Keeping pace with these developments has been computer-aided instruction utilizing various artificial intelligence technologies. The impacts of this growing capability on the stringent requirements for efficient and effective control center operations personnel is an area of much concentrated study.

This paper addresses the current research and development efforts in the area of automated tutoring systems for the spacecraft command/control environment being conducted by the Goddard Space Flight Center in conjunction with the Center for Man/Machine Studies at the Georgia Institute of Technology.

INTRODUCTION

Goddard's involvement with Intelligent Tutoring Systems (ITS) is coming about through an evolutionary system-upgrade process whose catalyst is the embedding of knowledge-based Operator Assistants (OA) in the ground systems for near-earth unmanned scientific satellites. Figure 1. depicts this evolutionary process. The paper will concentrate on the three major phases of this system evolution and will detail the architectural aspects of the intelligent tutoring concept being formulated. Since data from some initial experiments is still being analyzed the performance of a prototype ITS in a ground system environment will be the subject matter of future papers.

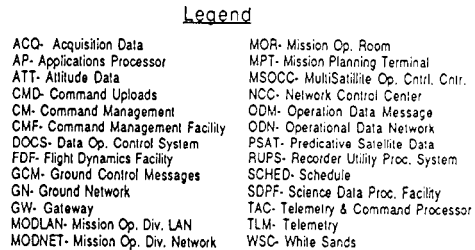


Evolution of C⁴ Environment

Figure 1.

C⁴ ENVIRONMENT

The environment in which we plan to embed intelligent tutoring systems is that which supports the command, control, computation and communication functions for near-earth unmanned scientific satellites. An abstracted view of this environment is presented in Figure 2.



ORIGINAL PAGE IS OF POOR QUALITY

way of representing how actions reflect cognitive processes. The model's structure should be a blend of a task performance and metacognitive model. We need to consider both the domain-specific characteristics of performance and domain-independent problem-solving behavior.

The basis for our operator's associate is the Operator Function Model (OFM). The OFM is a mathematical representation of how an operator might decompose the task of controlling a complex system into its simpler parts. The OFM is structured as a heterarchic-hierarchic network of finite-state automata. It is also a dynamic model, and this dynamic quality is represented as next-state transition functions that describe movement between states (nodes in the network). The OFM models operator-system interaction rather than the workings of the systems itself. The nodes in the network are operator actions, tasks, subfunctions, and functions. In particular, at the topmost heterarchic level are the major operator functions. Each function decomposes hierarchically into subfunctions, tasks and actions (either cognitive or manual). The next state transition functions can be modeled as system triggering events that cause the operator to switch to a different function, subfunction, or task. Figure 3 illustrates the OFM heterarchic-hierarchic framework.

The OFM specifies normatively how an operator should control the system. Given that the operator-system interaction is well-defined, an OFM can be constructed to model at least one reasonable method of control. Thus, the OFM is a prescriptive model in that it specifies non-deterministically a set of plausible manual and cognitive control actions, as well as goals and subgoals, given the current system state.

Operator behavior is prescribed in the context of the current state of the system. As a well-defined mathematical entity, the OFM is a computational model of human performance. The OFM is an operator model that is concerned with the operator-system interaction and the operator's functions within that interaction. Finally, the OFM is both a task and metacognitive model. It is a task performance model in that actual operator actions are mapped onto hypothesized tasks, and errors of omission or commission are clearly recognizable. It is also in part a metacognitive model that characterizes generally the decomposition of operator functions from goals to subgoals, function, and manual and cognitive actions. It combines the richness of a multi-leveled representation with the mathematical rigor necessary for implementation. Thus, the OFM, both conceptually and methodologically, is a suitable model for the basis of an intelligent aiding system.

After the successful application of the OFM in analyzing and developing a normative model of the subset of operator activities in the Multi-Satellite Operations Control Center (MSOCC) it became apparent that this modelling technique could be used to develop a knowledge base for an expert system whose function was to provide assistance to an operator. This expert system, named OFMspert, is an example of a type of intelligent system we call an "Operator Associate". To date an experimental version of OFMspert has been developed and demonstrated at the Center for Man-Machine Studies at Georgia Tech.

This line of research recognizes that today and well into the future the human operator is a

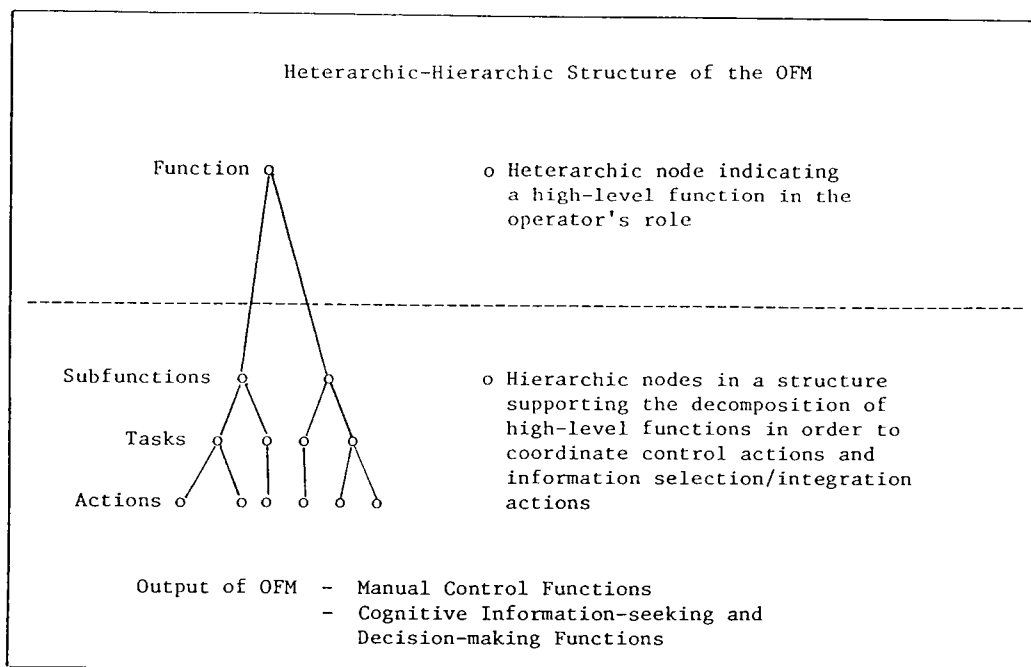


Figure 3.

critical component in control systems. The goal of an Operator Associate is to enhance and amplify the skills of the human operator and to exploit the strengths of all system components human or otherwise. The Operator Associate is designed to provide a dynamic symbiosis between the human operator and the rest of the control system.

The basic requirement for a viable Operator Associate is a normative model of the human operator behavior in various system contexts. For our system this model is provided by the OFM. As a basis the OFM provides information relative to the current operator state, predicted operator state and an assessment/predictor of the operator's goals, functions, intuition and performance.

In characterizing our version of Operator Associate two broad classes of operational capabilities or properties emerge, namely control and understanding. The control properties allow for the assumption, by the Associate, of varying levels of dynamic control of some part of the operational system. The level of control turned over to the Associate is determined by the human operator. The understanding properties provide the Associate with the capability of inferring current system goals, and offering context dependent assistance, advice and/or reminders to the human operator.

At the current time the focus of our development activity is on the understanding of the Associate. The level of understanding which can be supported by the Associate is, of course, a function of the application of the underlying operator model in explaining system operations.

Part of any understanding system are functions which we collectively call "intent inferencing". Intent inferencing tries to provide plausible explanations for observed operator actions given the current system state and past operator actions. Intent inferencing attempts to understand operator actions by interpreting them within the context of some normative model. In our case this normative model is provided by the OFM.

In providing information to support intent inferencing the heterarchic-hierarchical structure of the OFM comes into play. Briefly, the heterarchic nodes correspond to the high-level functions in an operator's role. Each such high-level function has associated with it a three-level hierarchy which supports a decomposition of the function into subfunctions, tasks, and actions. In the current implementation of the Associate this model information is manipulated by means of a blackboard system typical of those in use in current artificial intelligence systems.

Figure 4 depicts the blackboard model used in OFMspert. This model uses a three level hierarchy of knowledge sources (KS). The Strategy KS determines what type of event to focus on, the Activator KS selects a specialist KS appropriate for the selected event, and the Specialist KS provides the knowledge used to make modifications

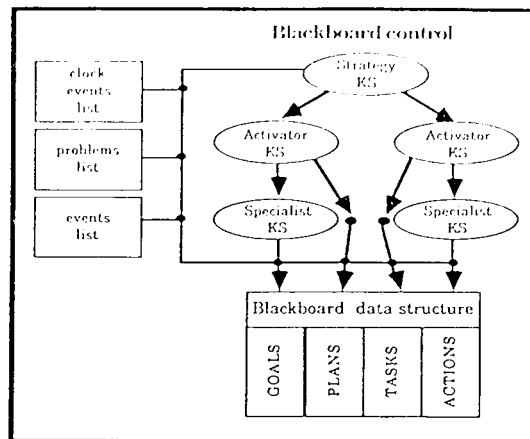


Figure 4.

to the blackboard. The current overall architecture for the OFMspert is shown in Figure 5. The Enhanced Normative Model with its Goals, Plan and Task (GPT) is based on the OFM.

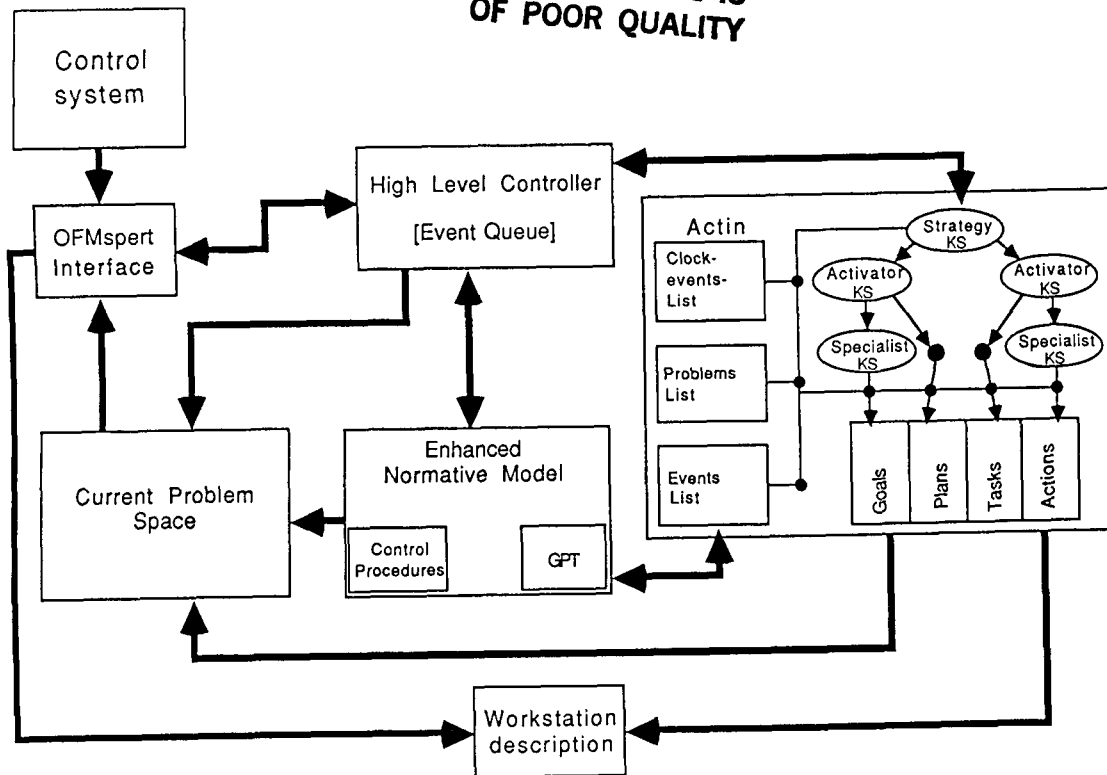
OFMspert is proposed as a conceptual foundation for the implementation of artificial intelligence in POCC ground control systems. The OFMspert does not automate a portion of the control system, but rather extends the capabilities of the human operator responsible for system operation. OFMspert is an architecture that provides the operator with an automated associate to which the operator can delegate routine tasks. The transfer of control from human to computer and back is accomplished smoothly and the computer-based associate is capable of inferring current system state and likely operator functions. Finally, OFMspert is capable of explaining what it is doing; explanations are given in the context of the OFM that defines both the operator and computer-based associate's role. These capabilities rest on the integrity of the OFM and the knowledge structures that comprise it. As such, OFMspert is an extension of the OFM models developed as part of this program with extensions in areas that are quite promising for C⁴ applications.

C⁴ - ITS

ITSSO (intelligent tutoring system for satellite operators) [6] is a design for a model-based, on-line tutoring system for operators of complex, predominantly automated dynamic systems. The design is illustrated in the context of GT-MSOCC. The purpose is to provide embedded training for novice operators learn how to supervise a complex, multifunction ground control system. This application is one with a great deal of practical appeal for current and future ground control applications.

Computer-based training is a natural application of the OFM for a dynamic system. The model dynamically specifies current operator functions related subfunctions, and both manual and cogni-

ORIGINAL PAGE IS
OF POOR QUALITY



OFMspert Architecture

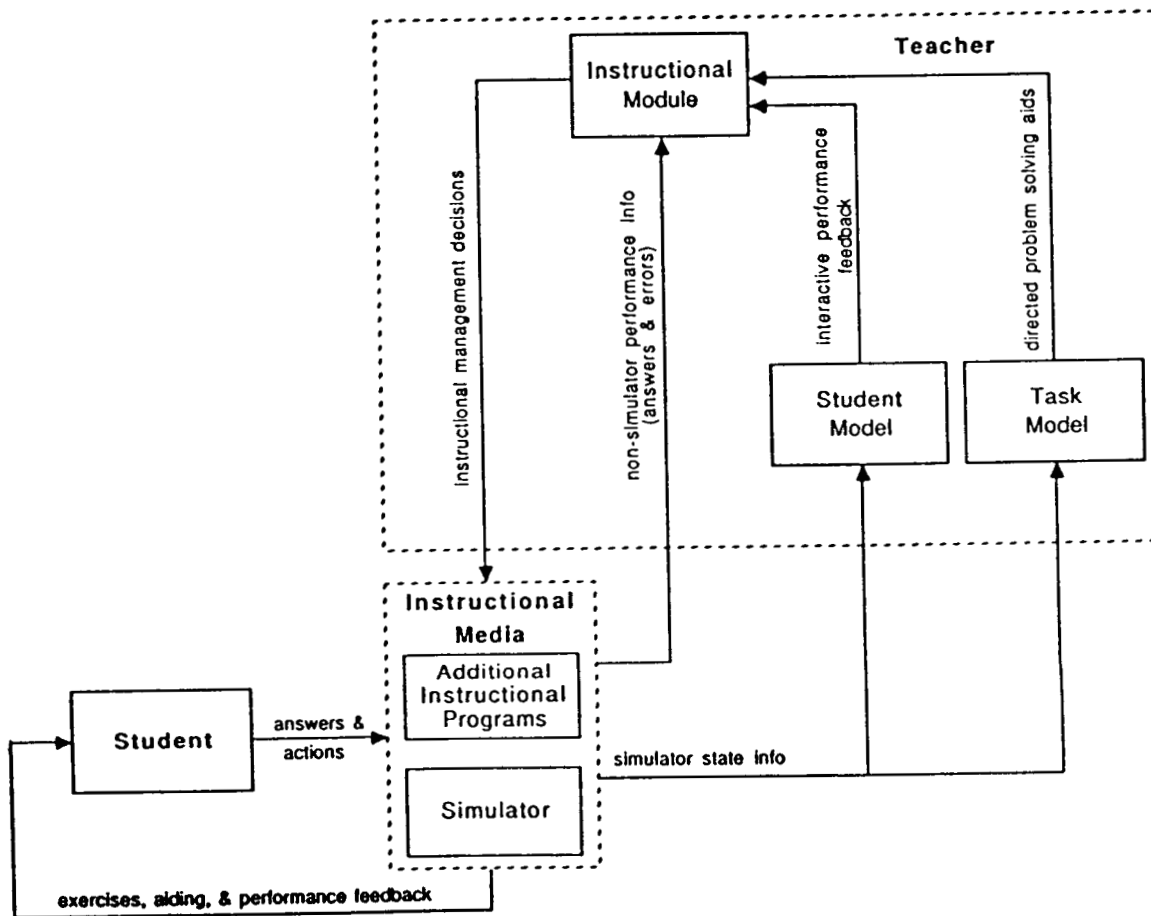
Figure 5.

tive actions. The OFM is a normative model and runs on-line. It interprets current system state and infers likely operator functions. As a result for novice operators, it could be quite useful in providing suggestions about what to do and how to do it given current system state.

A more interesting feature of the work is the intelligent and adaptive nature of the training system. In addition to simply providing suggestions, the OFM can be used to model the novice's current knowledge of the operator tasks. By examining the differences between the normative OFM and the student OFM, an on-line system tailors problems or scenarios within the context of GT-MSOCC to help the novice learn a given function or a procedure needed to carry out the function.

Essentially, the ITSSO research assumes that intelligent, on-line training or tutoring involves several components: a domain or task to be learned, e.g., GT-MSOCC operator operations, a model of the teacher (normative OFM), a model of the novice or student (descriptive OFM), and a set of teaching strategies. Teaching strategies allow the teacher model to modify the domain in order to teach the student new knowledge, increase experience or understanding of previously learned knowledge or procedures, and remediate error or misconceptions in the student's understanding of system or operator functions.

The current ITSSO concept is based on a tutoring system architecture established in [1]. Figure 6 displays this architecture.



Architecture for an Instructional System.

Figure 6.

One of the major basic accomplishments that has been achieved principally in the work conducted by our colleagues at Georgia Tech has been a characterization of the three major models, the task, student and instructional models, which form a basis of our ITS architectural concepts. The preliminary and high-level view of these models is being refined and detailed as we gain experience with the models.

The following tables, developed in [2], present a summary of the ITS architecture as it is currently envisioned.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

TABLE I SUMMARY OF ITS ARCHITECTURE

Assumptions:

The instructional program has access to a simulator of the large-scale, dynamic system on which the task to be learned is performed.

General Knowledge Representation Requirements:

1. Represent knowledge from multiple viewpoints
2. Represent knowledge at multiple levels
3. Represent knowledge at the correct grain size

Task Model

Function:

The function of the task model is to provide directed problem-solving aids.

Description:

The task model is a prescriptive model of the task, which specifies dynamically the relationship between goals, system states, and actions.

Requirements:

1. The model prescribes actions based on the current system state and current operator goals.
 - a. It must account for a dynamically changing system.
 - b. Suggested actions must be readily understandable by the student.
 - c. Suggested actions must be readily applicable by the student.
2. Students must be able to learn the the prescribed strategy.
3. Knowledge is organized as concepts and metaconcepts.

Knowledge Representation:

1. Concepts are represented as frames.
2. Metaconcepts are represented as production rules.

Student Model

Function:

The student model identifies correct and erroneous student actions and provides performance feedback.

Description:

The student model is a descriptive model of the student, which organizes student actions in the context of expert behavior.

Requirements:

1. The student model provides a method of organizing correct student actions.
2. It provides a method of organizing student errors.

Knowledge Representation:

1. Concepts are represented as frames with slots for correct and incorrect actions.
2. Metaconcepts are represented as production rules.
3. Error rules specify important or common errors.

Instructional Model

Function:

The function of the instructional module is to make instructional management decisions.

Description:

The instructional module contains a set of production rules for choosing the instructional medium, curriculum, pace of instruction, amount of feedback, and degree of control a student may exercise.

Requirements:

1. Instructional media
 - Alternate problem solving with quizzes and exercises
2. Curriculum
 - The order of presentation of material proceeds from easy to difficult
3. Pace
 - a. Present new concepts if none are to be reviewed.
 - b. Alternate new and review concepts.
 - c. Review concepts in which the number of errors exceeds the threshold value.
4. Feedback
 - a. Provide feedback during only part of the lesson.
 - b. When allowing feedback, present feedback immediately after every error.
 - c. Summarize feedback for a problem when a student requests it and at the end of a problem.

Knowledge Representation

1. Problems and exercises are contained within slots of the concept frames.
2. Rules make instructional management decisions.

ORIGINAL PAGE IS
OF POOR QUALITY

CONCLUSIONS

A prototype of ITSSO was implemented in the context of GT-MSOCC and empirically evaluated. Ten subjects, tutored by ITSSO, controlled the GT-MSOCC system. After training, the system performance of ITSSO-trained subjects was compared to that of subjects trained by human instructors. Data analysis is almost complete. Preliminary analysis indicates that subjects trained with ITSSO control GT-MSOCC as effectively as those trained one-on-one by a human instructor.

As the ITSSO concepts mature it is planned that ITSs will be introduced into the operational environments here at Goddard to provide additional support and training for the operators of the future.

The purpose of this paper has been to introduce the reader to emerging ITS-related research and development being sponsored by the Goddard Space Flight Center. Spurred on by the success we have experienced in introducing expert system technology into our C⁴ environments, the expansion of our knowledge-based technologies to include ITSs is proving to be a productive and focused application of experiences with artificial intelligence.

A major focus of our ITS-related research will be teaching strategies which can reflect changes in both the Student and Domain Models.

The reader who would like a more detailed view of our current work is urged to consult the references from which this article was developed.

ACKNOWLEDGEMENTS

The author wishes to sincerely thank Dr. Christine Mitchell and her associates at the Center for Man-Machine Systems Research at Georgia Institute of Technology for providing a rich research base to support continued work on ITSs.

Thanks also go to Robert Dominy, Troy Ames and Robert Dutilly at the Goddard Space Flight Center for helping to clarify the C⁴ Environment and to Dorothy Perkins and John Dalton for providing management support and encouragement.

REFERENCES

1. Mitchell, C. M., "GT-MSOCC: A Research Domain for Modeling Human-Computer Interaction and Aiding Decision Making in Supervisory Control Systems," *IEEE Trans., Systems, Man, and Cybernetics*, SMC-17, No. 4, July/August, pgs. 553-572, 1987.
2. Fath, J. L., "An Architecture for Adaptive Computer-Assisted Instruction Programs for Complex Dynamic Systems," Center for Man-Machine Systems Research, Georgia Institute of Technology, Report No. 87-3, June, 1987.
3. Rubin, K. S., Jones, Patricia M., and Mitchell, C. M., "OFMSpert: Inference of Operator Intentions in Supervisory Control Using A Blackboard Architecture," Center for Man-Machine Systems Research, Georgia Institute of Technology, Report No. 87-6, December, 1987.
4. Jones, P. M., "Constructing and Validating a Model-Based Operator's Associate," Center for Man-Machine Systems Research, Georgia Institute of Technology, Report No. 88-1, March, 1988.
5. Truszkowski, W., "The Operators Associate: An Operator Function Model Expert System," Research and Technology 1987, Goddard Space Flight Center, May, 1988, pgs. 207-209.
6. Resnick, D. E., "ITSSO: Intelligent Tutoring System for Satellite Operators," Center for Man-Machine Systems Research, Georgia Institute of Technology, Report No. 88-2, June, 1988 (in progress).

CONSERVATION OF DESIGN KNOWLEDGE

Larry Leifer
Fred Lakin
John Wambaugh
David Cannon
Stanford University Center for Design Research

Cecelia Sivard
Mark Sullivan
NASA Ames Research Center

(Paper not provided by publication date.)

Knowledge Acquisition for Case-Based Reasoning Systems

Christopher K. Riesbeck
 Dept. of Computer Science Cognitive Systems, Inc.
 Yale University New Haven, CT 06511
 New Haven, CT 06520

Abstract

Case-based reasoning (CBR) is a simple idea: solve new problems by adapting old solutions to similar problems. The CBR approach offers several potential advantages over rule-based reasoning: rules are not combined blindly in a search for solutions, solutions can be explained in terms of concrete examples, and performance can improve automatically as new problems are solved and added to the case library.

Moving CBR from the university research environment to the real world requires smooth interfaces for getting knowledge from experts. We describe the basic elements of an interface for acquiring three basic bodies of knowledge that any case-based reasoner requires: the case library of problems and their solutions, the analysis rules that flesh out input problem specifications so that relevant cases can be retrieved, and the adaptation rules that adjust old solutions to fit new problems.

Introduction

Case-based reasoning means reasoning from prior examples. A case-based reasoning system (CBRS) has a case library, containing 100s, 1000s or more cases. Each case describes a problem and a solution to that problem. The reasoner solves new problems by adapting relevant cases from the library.

Case-based reasoning is an alternative to rule-based reasoning. A rule-based reasoner has a large library of rules of the form, "IF A THEN B." These are chained together in various combinations to solve problems. A rule-based system will be flexible and produce nearly optimal answers, but it will be slow and prone to error. A case-based system will be restricted to variations on known situations and produce approximate answers, but it will be quick and its answers will be grounded in actual experience. In very limited domains, the trade-offs favor the rule-based reasoner, but the balance changes as domains become more realistically complex. Realistic domains involve a large number of number of rules, many of which are quite difficult to formalize properly, linked into long, tenuous chains of reasoning. With enough pre-stored solutions in a CBRS, there is almost always short path between the input case and the retrieved solution.

One important potential advantage of case-based reasoning is that human expertise seems more like a library of past experience than a set of rules. Hence cases should better support knowledge transfer (communication of expertise from domain experts to system) and explanation (justification of solution from system to domain experts). To increase the reasoner's knowledge, an expert would add

more cases. To explain why it came to the conclusions it did, the reasoner would show the cases from which it built its solution.

To make the utility of the case-based approach clear, consider a case-based reasoner that only retrieved cases similar to a given situation, but did no reasoning at all. For example, a tax advisor that showed me examples of tax forms filled out by people in circumstances very similar to mine would be very helpful, even though it didn't try to do my taxes for me. A tax expert could extend the system simply by adding more examples. The system in this case serves as a fairly direct conduit, transferring knowledge from expert to user.

This is the ideal situation, but there is one major problem that has to be solved when building a case library: indexing the cases so that the right cases are retrieved. The features that make one case similar to another are usually not explicitly in the input data, but are inferred using domain-specific knowledge. The domain expert has to add this knowledge to the system along with the cases.

Based on basic research in case-based reasoning [1], Cognitive Systems Inc. is developing a case-based reasoning shell whose primary function is to help a domain expert enter and organize a case library. In this paper we will describe the kinds of features that have been found useful for knowledge acquisition in a case-based reasoner.

A Case-Based Reasoning Shell

One way to look at a CBRS is to contrast it with a database management system (DBMS). The cases are like records in the database. (In fact, in application areas where databases exist, e.g., financial domains, cases really *are* records.) In a standard DBMS, records are retrieved with queries, such as "List all employee records where the salary field is over \$50,000 and the position field is not 'manager'." The DBMS retrieves all records that satisfy the constraints of the query. To do this, a database administrator in advance has organized the database, splitting different kinds of information into different files, and telling the DBMS to create indices for certain fields of certain records. For example, an employee database might be split into a file of employee names and ID numbers, indexed by names, a file of personal information about each employee, e.g., employee ID, home address, age, and so on, indexed by employee ID, and a file of payroll information, also indexed by ID. With modern query languages, the user doesn't have to know how information is actually split up, although these choices will make some queries more efficient than others.

A case-based reasoning system differs from a DBMS in two fundamental ways. First, a CBRS query is simply a

Figure 1: On-Screen Input Form ©Cognitive Systems, Inc. 1988

partially filled-in record, e.g., a tax form with personal and income information filled in, a loan application with the loan request and income information filled in, or a battle situation assessment form. The query is not a pattern of constraints, such as "greater than \$50,000", but a concrete description of some current situation. There is no query language to learn.

Second, the CBRS retrieves cases by *best partial match*. Retrieved cases usually do not match the input exactly. Instead, they are the cases that are closest to the input query, based on domain-specific information about what matters and what doesn't. In a Normal DBMS, records either match the query pattern or they don't. If a user wants to find records *similar* to a particular case, the user has to form a query pattern, replacing particular values with ranges of possible values. The user has to know what's in the database, what extra fields to calculate, how to replace particular input values with ranges, and so on. Example-based interfaces, such as Query-by-Example [2], make it easier to generate query patterns, but they still do not allow a user to simply enter a description of the current situation and let the system take care of everything else.

To use a CBRS, a user enters the current case, by filling out an on-screen form that, ideally, mimics exactly forms the user is already familiar with. Figure 1 shows an example of a battlefield intelligence estimate form that can be filled in on-screen in Cognitive System's shell-based battlefield advisor demo.

The CBRS application then retrieves forms describing similar cases. These retrieved forms contain additional fields, called *output fields*, indicating actions taken and outcomes observed in those previous situations. Armed with these exemplars, the user can make an intelligent choice about what to do in the current situation, and, optionally, add the new case to the library for future reference.

Knowledge Acquisition

To determine best matches, a CBRS has to know how to

- infer values for internal fields that a user would not be expected to input, e.g., the debt to income ratio in a loan application, or the attacker to defender ratio in a battle situation
- relate different field values to each other, e.g., incomes fall into brackets for tax purposes, and a prepared defense is closer to a fortified defense than it is to a hasty defense in a battle situation
- rank different fields as more or less important in determining best matches, e.g., income is more important than name for tax advice purposes, and strength ratios are more important than absolute sizes in assessing battles

The job of the Domain Expert Interface component of the Case-Based Reasoning Shell is to enable a domain expert, with little or no programming knowledge, to add these three kinds of knowledge, plus cases, of course, to form a *case library* that, in conjunction with the Shell, forms a CBRS application usable by end users.

The Domain Expert Interface, by necessity, is somewhat more complicated than the end user interface, but, by using ideas from DBMS interfaces and taking advantage of the concrete, real-world-based nature of case-based reasoning, it is still possible to have an interface that is much simpler and easier to use than those commonly found in rule-based expert systems.

For example, adding new cases to the case library is simply a matter of filling out the same forms that the end user sees. The only difference is that the domain expert also fills in the output fields. For example, in the battle assessment domain, the domain expert would fill in not only the initial battle situation, but also the battle outcomes, post-battle analyses of why things happened as they did, graphical annotations, and so on. If an on-line database already exists for a domain, the shell can convert the database records to cases, so that the domain expert just has to add any output fields not included in the original database, and organize the cases, as described below. For example, the Quantified Judgment Model (QJM) battle database, developed at Data Memory Systems Incorporated, was used to initialize the case library for Cognitive Systems' battlefield advisor.

Cognitive Systems' CBR Shell provides the domain expert with two interfaces to assign relative importances to fields in a case. One interface allows the expert to "color" the fields of the input form, where each color represents a different level of importance. The other interface, present when different cases are being compared on-screen, lists fields in order of importance, and allows the expert to drag fields up or down, to fine-tune the relative rankings. Furthermore, the Shell can assign an initial set of importance levels to fields, by looking at which cases have similar values in their output fields. Our initial experience suggests that field importances are hard for experts to determine, and not as robust for determining case similarities as the other two kinds of information described next. The initial values assigned by the Shell are usually best left as is, except in special circumstances.

To specify how different field values relate to each other, e.g., to allow the expert to say that a prepared de-

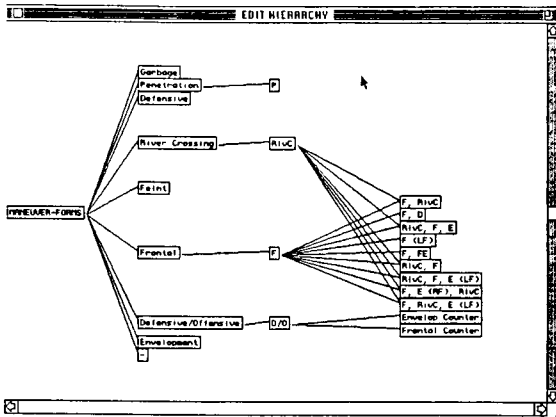


Figure 2: Abstraction Editor Screen ©Cognitive Systems, Inc. 1988

fense is closer to a fortified defense than to a hasty one, the expert uses a graphical *abstraction hierarchy editor*, similar to those found in other AI shells such as KEE and ART [3]. Each field in a form has its own hierarchy of values, since the values, especially if they are numbers, might mean different things in different fields. The hierarchy is initialized to a simple list of all the values seen in that field in the entire case library. The domain expert groups these values together to form the hierarchy. Thus, in the field for defensive posture, the expert might group prepared and fortified defenses together as strong defenses, and group strong and hasty defenses together as defenses. The closer two values are grouped, the better they are considered to match when the CBRS is looking for similar cases. Figure 2 shows an abstraction hierarchy screen for Cognitive's battlefield advisor.

Finally, to specify derived fields, e.g., to create a field that is the attacker to defender strength ratio, the CBR shell uses a graphic formula building interface similar to that found in database systems such as Double Helix [4]. The expert selects various fields from the case form and links them to arithmetic, comparison, and other kinds of operation icons. Such an interface avoids problems with syntactic errors, and lets the domain expert know what options are available at any time for constructing a formula. Figure 3 shows a formula screen for Cognitive's battlefield advisor.

The usefulness of a CBRS of course depends on how well it actually does in find similar cases. Therefore, the center of interaction in the Domain Expert Interface is a screen that displays, for a given test input case, the five best matches that currently exist, given the current importances, hierarchies, and derived fields. The display uses a "compare and contrast" columnar format listing the output fields, followed by the fields ranked as most important. If the expert sees cases being retrieved that he or she considers very dissimilar to the input case, the expert can tell from this display if truly similar cases are not matching because certain field values are poorly grouped in their hierarchies, or if an important combination of values is not being calculated, or if some field of information is simply

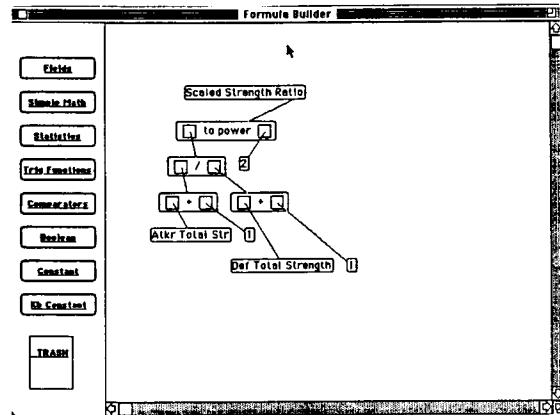


Figure 3: Formula Editor Screen ©Cognitive Systems, Inc.
1988

Case Viewer: QM battle databases / Fifth View

Field Name

Input Case

Best Five Cases

Hypothesis #3 4080: Triforce 4230: Apollo 4420: Complex 4490: Villa Ora 5040: Berceuse

☐ 4080

☐ 4230

☐ 4420

☐ 4490

☐ 5040

Cases:

Success (Victor)	ROOT	A	B	A	B	A	B	A	B
Defender	30w 31st B	30w Harrier	30w 3d Fz	30w 3d Fz	30w 3d Fz	30w 3d Fz	30w 3d Fz	30w 299th	
Def Resolution	ROOT	MD	MD	S	S	N	S	MDL	
Def Mission Accomplish	0	2	2	0	0	0	0	4	
Attacker	US 62d Inf	US 3d Inf	30w 1st Inf	30w 1st Inf	US 34th In	30w 1st Inf	30w 1st Inf	30w 1st Inf	
AttR Resolution	ROOT	P	P	P	P	P	P	8	
AttR Mission Accomplish	0	7	0	0	0	0	0	7	
Comment		Warning:	Comment:	None:	Comment:	Comment:	Comment:	Comment:	

Importance 6:

Total Power Ratio	0.18	0.28	1.02	1.41	5.17
Scalad Strength Ratio	1.71	0.49	0.22	1.38	1.72
Logistics Capability	N/-	C	C	C	C
Leadership	C	C	C	C	C
Initiative	N/-	N/-	N/-	N/-	N/-
Force Quality	N	N	N	N	N
Force Preparedness	N	N	N	N	N
Depth	N	N	N	N	N

Importance 3:

Scalad Armor Ratio	1	21.64	2.35	0.60	2.05	191.60
Weather	DST	DST	DST	DSC	DST	DST
Training & Experience	C	C	N/A	N/A	C	C
Terrain	FM/FM	FM	FM	FM	FM	FM/FM

Figure 4: Best Match Comparison Screen ©Cognitive Systems, Inc. 1988

missing from the database. Figure 4 shows a compare and contrast screen for Cognitive's battlefield advisor.

Conclusions

This paper has described the basic features of a knowledge acquisition interface for a case-based reasoning shell. The shell uses a form-filling metaphor for case entry, and a graded match mechanism for displaying case similarity. A domain expert organizes the library by specifying what fields in a form are important, how well different values of a field match each other, and what additional derived fields have to be calculated to capture important non-input features. The result of the domain expert's efforts is a case-based reasoning application that can take an input situation and retrieve relevant cases from the case library to assist a human decision maker or problem solver.

Acknowledgments

Basic research on case-based reasoning was funded in part by the Air Force Office of Scientific Research under con-

tract F49620-82-K-0010. Application of Cognitive System's Case-Based Reasoning Shell to the battlefield assessment domain is being funded in part by the Defense Advanced Research Projects Agency. The battlefield advisor was built with Cognitive Systems' case-based reasoning shell, using Data Memory Systems' Quantified Judgment Model battle database. The screens shown in the figures are copyrighted by Cognitive Systems, Inc.

References

1. Kolodner, J., editor, *Proceedings of the First Case-Based Reasoning Workshop*, Morgan Kaufmann Publishers, Inc., Los Altos, CA., 1988.
2. Ullman, J. D., *Principles of Database Systems, Computer Software Engineering*, Computer Science Press, Rockville, MD, second edition, 1982.
3. Gevarter, W., The nature and evaluation of commercial expert system building tools, *IEEE Computer*, 20(5), May 1987, pages 24-41.
4. Hirschberg, G., Double helix or nothing, part 1, *MacUser*, 4(4), April 1988, pages 108-116.

PSYCHOLOGICAL TOOLS FOR KNOWLEDGE ACQUISITION

Henry H. Rueter
Vector Research, Incorporated
P. O. Box 1506
Ann Arbor, Michigan 48106

Judith Reitman Olson
School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109-1234

Knowledge acquisition is said to be the biggest bottleneck in the development of expert systems. The problem is getting the knowledge out of the expert's head and into a computer. In cognitive psychology characterizing mental structures and why experts are good at what they do is an important research area. Is there some way that the tools that psychologists have developed to uncover mental structure can be used to benefit knowledge engineers? We think that the way to find out is to browse through the psychologist's toolbox to see what there is in it that might be of use to knowledge engineers.

Expert system developers have relied on two standard methods for extracting knowledge from the expert: (1) the knowledge engineer engages in an intense bout of interviews with the expert or experts, or (2) the knowledge engineer becomes an expert himself, relying on introspection to uncover the basis of his own expertise. Unfortunately, these techniques have the difficulty that often the expert himself isn't consciously aware of the basis of his expertise. If the expert himself isn't conscious of how he solves problems, introspection is useless.

Cognitive psychology has faced similar problems for many years and has developed exploratory methods that can be used to discover cognitive structure from simple data.

We will skip over what we call "direct" methods for knowledge acquisition. Direct methods include interviews, questionnaires, protocol analysis, interruption analysis, and inferential flow analysis. Our goal here is to expose the reader to "indirect" methods, methods which are likely to be a good deal less familiar to the practicing knowledge engineer: multidimensional scaling, hierarchical clustering, general weighted networks, ordered trees, and repertory grid analysis. But first, a few points need to be made about the variety of ways that experts can organize information.

Simple OBJECT-ATTRIBUTE-VALUE triples and forward and backward search strategies are only a few of the knowledge structures and search strategies that human experts seem to have. Expertise is primarily a skill of recognition, of "seeing" old patterns in a new problem. Chess experts, for instance, have the same limited abilities as novices to hold information for analysis; their non-chess memory abilities are not exceptional. They excel because they have hundreds of thousands of chess configurations in their heads, and because they can quickly encode the current situation into constellations of previously-seen chess patterns. The choice of candidate good moves for the expert is thus restricted to a small set of know-good moves that fit the patterns, whereas the novice has no such expert pattern knowledge to filter out bad candidates. [1, 2]

There is also evidence to suggest that experts see more richly encoded patterns than novices do. They have organized the concepts in their knowledge bases with much more depth and with many more central associations than novices. For example, in the laboratory we found that expert ALGOL programmers had much more structure in their concept relationships than did novice programmers. Furthermore, the experts' mental organizations were highly similar, whereas the novices had scattered, idiosyncratic organizations for ALGOL-specific concepts. [3]

Not only do experts have information organized in a highly structured way, they also use a variety of different kinds of knowledge structures. For instance, some things are stored in simple lists like the months of the day and the days of the week. Other information fits a table better, information such as calendar appointments and the periodic table. Some information is stored as a flow diagram, such as decision trees, for example, representing the routing of telephone messages to people who can handle them. There is information stored

in hierarchies of relationships, nested categories or clusters, such as animal taxonomies or familial relationships. Networks store richly connected language associations. Information concerning room arrangements or maps may be stored as a physical model or physical space. And, some information may be stored about a device's internal components and how they are causally related as a physical model, commonly referred to as a mental model. Thus, experts may hold what they know about objects and their relationships in many different representations, each suitable for a particular kind of reasoning or retrieval.

METHODS FOR KNOWLEDGE ACQUISITION

There are two general classes of methods for revealing what experts know. "Direct Methods" ask the expert to report on knowledge that can be directly expressed. This set of methods includes interviews, questionnaires, simple observation, thinking-out-loud protocols, interruption analysis, and inferential flow analysis. In contrast, "Indirect Methods" do not rely on experts' abilities to articulate the information that is used, or how it is used. Instead, indirect methods use other kinds of behavior, such as recall from memory or rating scales, as the basis for inferences about what the expert must have known (and, perhaps, the form in which it must have been represented) in order to produce the responses that were observed. Indirect methods include multidimensional scaling, hierarchical clustering, general weighted networks, ordered trees, and repertory grid analysis.

INDIRECT METHODS

All of the direct methods mentioned above ask the expert directly what he knows. They rely on the availability of the information to both introspection and articulation. Of course, it is not always the case that the expert has access to the details of his mental processing. In fact, it is not uncommon for experts to perceive complex relationships or come to sound conclusions without knowing exactly how they did it. In these cases, indirect knowledge elicitation methods are required.

In the following methods, experts are asked not to express their knowledge directly. Instead they are given a variety of other tasks, e.g., to rate how similar two given objects are, or to recall a collection of objects several times from prescribed starting points. From the results, the analyst then infers the underlying structure among the objects rated or recalled. All the indirect methods discussed here have been validated in experimental studies that have convincingly demonstrated their psychological validity.

To make progress, these different techniques must make assumptions about the way the data were produced. Assumptions must be made about

the nature of the mental representation: Is it physical space, lists, networks of association, tables, etc.? Furthermore, the stronger the assumptions that the analyst is able to make, the stronger the conclusions that can be made. Thus, it is important for the analyst to make a good guess about what form the expert's underlying representation is likely to take. An informed guess can be made after initial interviews with the expert, as well as from careful questioning and noting of object names and notations that the expert uses.

Of the methods to be discussed, multidimensional scaling, hierarchical clustering, and general weighted networks are the most general, in the sense that they make the weakest assumptions about the data being analyzed. These three methods can be reasonably applied to any similarity judgments, while repertory grid analysis and ordered tree analysis make strong psychological assumptions about the kind of mental structure and processes under investigation.

Multidimensional Scaling

Multidimensional scaling (MDS) is a technique that should be used only on similarity data that can be assumed to have come from stored representations of physical n -dimensional space [4]. The subject provides similarity judgments on all pairs of objects in the domain of interest. These judgments are assumed to be both symmetric and graded. This means that the similarity of A to B must be the same as the similarity of B to A (symmetry) and that there must be a continuum of possible similarity values relating A and B, not merely a simple judgment of similar or dissimilar (gradedness).

A computer program is required to perform the multidimensional scaling analysis. The result is a configuration of the objects in space. The dimensionality of the space and the metric that obtains in it are selected by the analyst, usually on the basis of trial and error.

Of course, it may not be possible to find a configuration that exactly represents the generating similarity matrix. In fact, each MDS solution has a "stress" value associated with it that provides a measure of the degree to which the computer-produced configuration and the input matrix differ. In practice, the analyst looks for the lowest stress solutions with the fewest dimensions.

The MDS technique is good for producing a diagram that the expert can later inspect and describe in more detail. It can reveal interesting clusters of objects, neighbor relations, and outlier, or "fringe" objects. One difficulty with this technique, as well as the others that we describe that require a similarity matrix, is the tedious and time-consuming process of collecting the pair-wise judgments. For n objects $n(n-1)/2$ judgments are

required, a number that soon grows quite large, even for motivated subjects.

A second difficulty with the technique is discovering the right space: in particular, the right dimensionality, the right distance metric, the right starting configuration, and the right interpretation of clusters and dimensions. Once the data are in hand performing the analysis is fairly straightforward, but interpreting the results requires some expertise.

Cluster Analysis

Like MDS, cluster analysis starts with a matrix of symmetric similarity judgments. There are many clustering algorithms, developed for many purposes, but for psychological investigations Johnson hierarchical clustering is the method of choice, because the result of this clustering technique is sensitive only to the ordinal properties of the similarity judgments and not to magnitude [5]. This insensitivity to judgment magnitude reflects the prudence required in interpreting psychological judgments.

Johnson hierarchical clustering produces hierarchical representation of the items of interest; the hierarchy takes the form of a rooted tree in which the items are the "leaves." Each subtree forms a cluster and the path that connects two items in the tree is a measure of the diameter of the smallest cluster that contains them both.

Hierarchical clustering is ordinarily done using either the "minimum" method, in which the similarity between two clusters is that of the most similar items in either, or the "maximum" method, in which the similarity between two clusters is that of the least similar items in either. The minimum method tends to give long, stringy clusters, the maximum method tight, spherical ones.

General Weighted Networks

This is a third method using a symmetric similarity matrix obtained from experts' pair-wise similarity judgments. In this case there is a somewhat more theoretical basis for the analysis: We assume that in producing the judgments the expert is traversing some mental network of associations, a network in which there is a single primary path between every two items, and, for some of them, a differently encoded, secondary path as well.

The object of this method, which was developed by Schvaneveldt, et al. [6, 7], is to reconstruct the associative network through the similarity judgments. In attempting this, Schvaneveldt, et al., recently investigated the nature of expertise in airplane pilot performance using networks.

The method requires a computer and works as follows: First, a Minimal Connected Network (MCN) is formed by connecting the most similar items, then the next most similar items, etc., with arcs until there is a unique path between any two items (a minimal spanning tree). In the second stage of the analysis more links are added to the MCN to form the Minimal Elaborated Network (MEN). To form the MEN we add a link between two items to the MCN if and only if it is shorter than the path between them in the MCN.

The interpretation of the MCN and MEN involves looking for:

- (1) dominating concepts—those that have a large number of connections to many other nodes; and
- (2) members of cycles—collections of items that are fully linked in circular paths.

In their exploration of the MCN and MEN for both expert and novice pilots Schvaneveldt, et al., collected similarity judgments on a set of flying terms having to do with "split-plane concepts." The analysis of the judgments revealed:

- (1) Expert's structures are simpler than students'.
- (2) Elaborated links connected larger integrated conceptual structures.
- (3) Experts could easily identify link relations in the networks, relations such as "affects," "is-a," "desirable," and "acceptable."

The fact that the experts were so clearly different from novice fliers suggests that this GWN technique can reveal significant aspects of expertise, aspects that clearly should be encoded into an expert system.

The object of the ordered tree technique is to induce a subject's mental structure for the set of to-be-recalled items from his recall orders. The structure will be an ordered tree, that is, a tree which reflects the subject's clustering and prioritization of the items of interest.

Unlike hierarchical clustering, the ordered tree technique is based on a detailed psychological model of how the recall orders are produced by the subject: It assumes that people recall all items from a stored cluster before recalling items from another cluster. (This is the hypothesis implicit in the concept of "chunks" in memory.)

This assumption builds on data from people recalling from known (learned) organizations.

Regularities found throughout a set of orders are taken as evidence of responsible mental structure and processing. Sets of orders need not come from recall; they can be obtained simply by asking subjects to order items so that items that are related are placed together.

The computer program that conducts ordered tree analysis examines all orders for sets of item that form connected suborders. The set of all such connected item sets forms a lattice of chunks, where the elements of the lattice are ordered by set inclusion. The lattice is converted into an ordered tree structure in which a node may be marked as unidirectional (only one order of its constituents was seen), bidirectional (only one order and its inverse), or nondirectional (more than two orders observed). The program can also perform certain advanced analyses in addition, such as calculating an index of organization and looking for anomalous, or "outlier," orders, whose exclusion from the analysis yields a new tree structure with significantly more structure.

This technique has been used in a variety of studies of expert-novice differences. In [3], for example, novice, intermediate, and expert ALGOL-W programmers were asked to recall ALGOL keywords many times from many different starting points while their performance orders were recorded. Experts differed remarkably from the novices. They showed much more organization, and the similarity among the expert structures (ordered trees) was far greater than that among the novices. In [21], furthermore, the pauses between recalls of successive items was accounted for by the number of chunk boundaries crossed in the inferred memory organization. There have been a variety of studies that have used this technique to reveal organization in different domains of expertise; all have shown a convergence among experts in their mental organization of the concepts.

Repertory Grid Analysis

This technique as used in [9] is the most integrated cognitive tool for knowledge acquisition of those presented here. It includes an initial dialog with the expert, a rating session, and analyses that both cluster the objects and the dimensions on which they were rated. Essentially, it is a free-form recall and rating session in which the analyst makes inferences about the relationships among objects and the relatedness of the dimensions that the expert finds important.

Since the use of repertory grid analysis as an expert system-building tool is beautifully covered in [9] we will give only a brief outline here, referring the reader to [9] for details.

Repertory grid analysis is a technique that comes from personal construct theory, a clinical tool intended to reveal the structure of a patient's emotional system. As used in knowledge engineering the first step in the analysis is an open interview with the expert, in which some important objects in the domain of expertise are elicited. Once a set of items is available, the analyst picks sets of expertise are elicited. Once a set of items is available, the analyst picks sets of three elements and asks "What trait distinguishes any two of these objects from the third?" The expert-supplied trait identifies a "dimension" in the domain. Then the expert is asked to rate all three objects along the named dimension. This process of asking for salient dimensions for further triples continues until the analyst is satisfied that the major dimensions of the system have been uncovered.

The analyst now constructs a matrix, or grid, with objects labeling columns and dimensions labeling rows. Then the expert is asked to fill in all the missing values, so that all objects are rated on all dimensions.

It is now possible to perform a cluster analysis on both objects and dimensions, using an appropriate similarity measure between the vectors of interest. Such analyses are used to identify prototypical dimensions and items.

CONCLUSION

Just as a statistician makes judgments about the suitability of a data set to the assumptions of a proposed analysis, the knowledge engineer must make judgments of the suitability of a method for knowledge elicitation to the kinds of knowledge the expert is assumed to possess. There are a number of ways these techniques can be misapplied for scientific discovery of mental organizations. However, if used as exploratory tools, these techniques can bring a great deal of information to the knowledge engineer [10]. With them, knowledge engineers can hope to uncover more of what experts know than can be learned through interviews or introspection.

REFERENCES

- [1] Chase, W. G., and Simon, H. A. "The Mind's Eye in Chess." In W. G. Chase (Ed.), VISUAL INFORMATION PROCESSING. New York: Academic Press, 1973.
- [2] DeGroot, A. D. THOUGHT AND CHOICE IN CHESS. Paris. Mouton Co., 1965.
- [3] McKeithen, K. B., Reitman, J. S., Rueter, H. H., and Hirtle, S. C. KNOWLEDGE ORGANIZATION AND SKILL DIFFERENCES IN COMPUTER PROGRAMMERS. Cognitive Psychology, 1981, 13, 306-325.

- [4] Shepard, R. N., Romney, A. K., and Nerlove, S. B., MULTIDIMENSIONAL SCALING: THEORY AND APPLICATIONS IN THE BEHAVIORAL SCIENCES. Vol. I New York: Seminar Press, 1972.
- [5] Johnson, S. C. HIERARCHICAL CLUSTERING SCHEMES. Psychometrika, 1967, 32, 241-254.
- [6] Schvaneveldt, R. W., Durso, F. T., and Dearholt, D. W. Pathfinder: "Scaling with network structures." Memoranda in COMPUTER AND COGNITIVE SCIENCE, Report No. MCCS-85-9, Computer Research Laboratory, New Mexico State University, 1985.
- [7] Schvaneveldt, R. W. Anderson, M., Breen, T. J., Cooke, N. M., Goldsmith, T. E., Durso, F. T., Tucker, R. G., and DeMaio, J. C. STRUCTURES OF MEMORY FOR CRITICAL FLIGHT INFORMATION. Air Force Human Research Laboratory, Technical Report 81-46, 1982.
- [8] Reitman, J. S., and Rueter, H. H. "Organization Revealed by Recall Orders and Confirmed by Pauses." COGNITIVE PSYCHOLOGY, 1980, 12(4), 554-581.
- [9] Boose, J.H. EXPERTISE TRANSFER FOR EXPERT SYSTEM DESIGN. New York: McGraw Hill, 1986.
- [10] Hart, A. KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS. New York: McGraw Hill, 1986.

AUTOMATED KNOWLEDGE ACQUISITION OF DESIGN RATIONALE

Joseph W. Sullivan
Jon Schlossberg
Bill Mark
Lockheed Missiles and Space Company

Mike Genesereth
Narinder Singh
Stanford University

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

NASA'S CREW EMERGENCY RETURN VEHICLE (CERV)
FOR THE SPACE STATION

Captain Brian Kelly
NASA Johnson Space Center

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

LIGHT WEIGHT ESCAPE CAPSULE FOR FIGHTER AIRCRAFT

Capt James A. Hubert
 Air Force Wright Aeronautical Laboratories
 AFWAL/FIER
 WPAFB, OH 45433-6553
 513-255-4008

ABSTRACT

Emergency crew escape capabilities have been less than adequate for fighter aircraft since before WW II. From the over-the-side bailout of those days through the current ejection seat with a rocket catapult, escaping from a disabled aircraft has been risky at best. Current efforts are underway toward developing a high-tech, "smart" ejection seat that will give fighter pilots more room to live in the sky but, an escape capsule is needed to meet current and future fighter envelopes. Escape capsules have a bad reputation due to past examples of high weight, poor performance and great complexity. However, the advantages available demand that a capsule be developed. This capsule concept will minimize the inherent disadvantages and incorporate the benefits while integrating all aspects of crew station design. The resulting design is appropriate for a crew station of the year 2010 and includes improved combat acceleration protection, chemical or biological combat capability, improved aircraft to escape system interaction, and the highest level of escape performance achievable. The capsule is compact, which can allow a reduced aircraft size and weighs only 1200 lb. The escape system weight penalty is only 120 lb higher than that for the next ejection seat and the capsule has a corresponding increase in performance.

BACKGROUND

Emergency crew escape capabilities have been less than adequate for fighter aircraft since before WWII, when over-the-side bailout was the only means of escape. The development of jet aircraft was accompanied by ejection seats that were catapulted from the cockpit. This was followed by the addition of a rocket for tail clearance and runway ejections and then a drogue parachute for stabilization and deceleration at high speeds. The current USAF ejection seat, the ACES II, includes a small gimbaled rocket that helps stabilize the escape system and airspeed sensors to vary parachute sequencing. From 1957 to 1984, the rate of major injury or fatality (M/F rate) for non-combat ejections with sufficient altitude above the ground was an average of 26 percent. The ACES II seat shows significant improvement with a M/F rate of 14 percent. When

this data is filtered to isolate the effect of airspeed, the results are very interesting. In fact, only 10 percent of the non-combat ejections were over 400 KEAS (knots equivalent airspeed, 687 psf dynamic pressure). Data from combat missions in Vietnam showed that ejection speed increased dramatically, with approximately 50 percent of ejections occurring at over 400 KEAS. Due to a limited amount of combat data, the non-combat data with known airspeed and cause of injury was used for judging injury rates. The M/F rate for ejections under 400 KEAS became 23 percent and over 400 KEAS was 65 percent. (The corresponding ACES II M/F rates are 9 percent and 70 percent.) Based on these rates, the combat ejection M/F rate could exceed 45 percent due to airspeed. Technology is currently available for the development of a controllable ejection seat under the Crew Escape Technologies (CREST) program. This program will demonstrate an escape system that can remain stable at speeds up to 700 KEAS (1660 psf) and steer away from the ground during low altitude ejections. An ejection seat based on the CREST program results will improve low altitude escape performance and provide greater protection at high speed. However, it will be difficult for an open ejection seat to meet the 700 KEAS goal while fighter aircraft can already fly at 800 KEAS (2100 psf) or more. The desire for further improvements in safe escape led to an effort to develop an escape capsule that could take advantage of current and emerging technology and perhaps become available early in the next century.

Escape capsules provide protection from the elements and are a natural solution to the high-speed escape problem. However, previous capsule experiences in the USAF have led to a generally bad reputation for capsule escape systems. The two operational capsules that have been flown (the F-111 and the B-1A, the B-1B has ejection seats) were based on technologies that are 20 years old or more (Figure 1). This lack of technological capability led to designs that were heavy and difficult to make feasible. The F-111 capsule now weighs 3,300 lb (crew of two), includes a large portion of the forward fuselage and contains many heavy instruments and controls. The B-1 capsule weighed nearly 10,000 lb (crew of 6) which resulted in a similar weight per crew member to that of the F-111 (about 1,700 lb). In contrast, an ejection seat with the capabilities sought by the

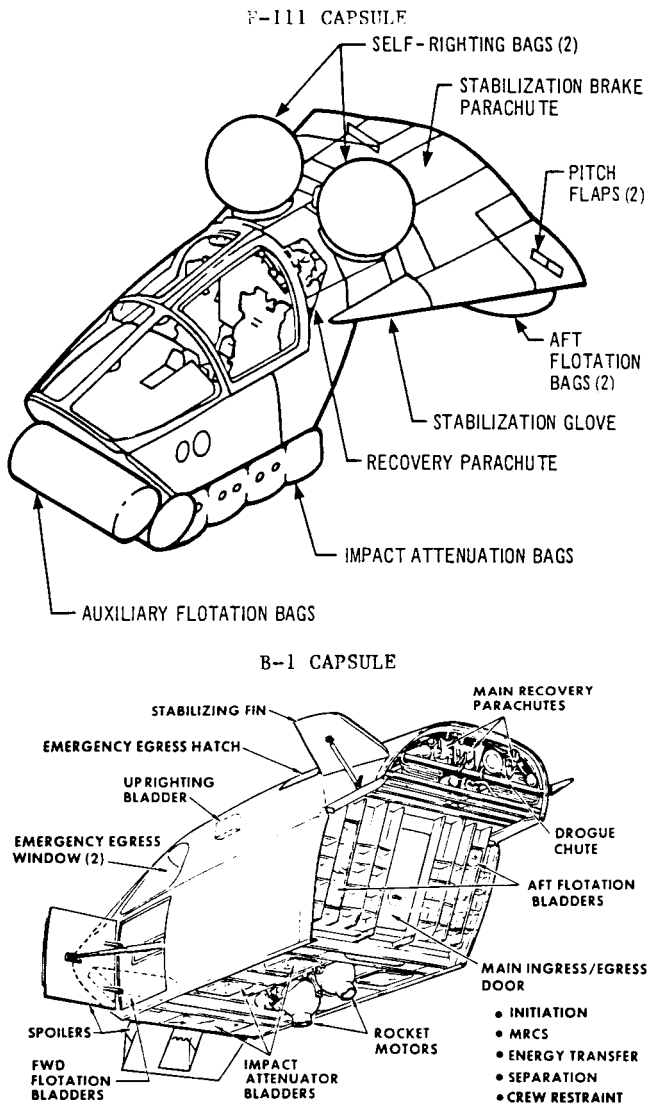


Figure 1. F-111 and B-1 Capsules

CREST demonstrator escape system is expected to weigh from 600 to 700 lb for one person when ejected. The comparatively higher weight of capsules leads to greater penalty to the aircraft and, because of aircraft weight limitations imposed, it is difficult to achieve the same performance levels as those possible for an ejection seat. The capsule weights are high for a number of reasons. The underlying reason is that the escape system designs were constrained by predetermined fuselage structural designs and crew stations. This led to excess volume in the capsules which allowed capsule weight increases caused by other aircraft systems. This approach also precluded the use of an insertable capsule which would reduce the amount of aircraft structure carried in the capsule and allow a minimum volume for the capsule. Lack of today's technologies prevented solutions to the problem of crew station/capsule weight as well. The older crew stations were full of control panels with their associated boxes, control units, computers and

countless wires. These combined weights added to the total that the escape rocket system had to accelerate. In addition, relatively dumb rockets and control systems were used that lacked efficiency and generally ended up undersized due to weight growth of the capsules. Another problem with the previous capsules was the method of landing them after an ejection. The most efficient approach at the time was to use inflatable airbags to absorb landing impact during parachute descent. This approach has a limited performance envelope which has led to a 15 - 20 percent major injury rate due to landing impact for the F-111. A factor that added to the weight problem and created maintenance difficulties was that the F-111 and B-1 capsules were integral to their aircraft fuselage and used explosive shaped charges as the means of separating for ejection. This meant that all capsule subsystems were accessible only through the skin on the fuselage. Periodic refurbishment of the capsule involved removing much of the skin and replacing all pyrotechnic components. This scarred past for escape capsules has severely limited investment toward future capsule escape systems.

Two other areas of concern are G-induced loss of consciousness and ingress and egress in a chemically or biologically contaminated (CB) environment. Today's fighter aircraft are designed to be able to turn at 9 G while fighter pilots can only withstand 7 to 9 G through intensive physical efforts. Also, these aircraft are capable of reaching acceleration levels of 9 G faster than pilots' bodies can compensate for them. This situation has led to 7 deaths of Air Force pilots directly attributed to G loss of consciousness since 1983. Finally, there is no current method for ingressing or egressing the cockpit in a CB environment while keeping the cockpit "clean". Efforts are underway to develop ways to keep the environmental control system air free of contamination through the use of catalytic converters or a closed loop system, but the pilot must be allowed in and out of the crew station. The current approach uses a dirty cockpit and the pilot must wear cumbersome, hot protective gear while flying the mission.

DISCUSSION

The Air Force Wright Aeronautical Laboratories, Aircrew Protection Branch has engineered an approach to providing capsule escape over the last four years with encouraging results. The solution involves integration of capsule, crew station, airframe, and crew member requirements and emphasizes the need for an independent crew station and escape system design group whose requirements must be included in future aircraft development programs. The program focused on providing high speed escape capability with maximum escape system performance, protection from aircraft combat accelerations and ingress and egress in a CB environment.

The escape capsule design that emerged from our effort known as Concept Development of a Canopy Escape Module was based on the F-16 geometry as shown in Figure 2. The F-16 has a large, single-piece transparency that is convenient for attaching a crew station to form a capsule. Also, there

ORIGINAL PAGE IS OF POOR QUALITY

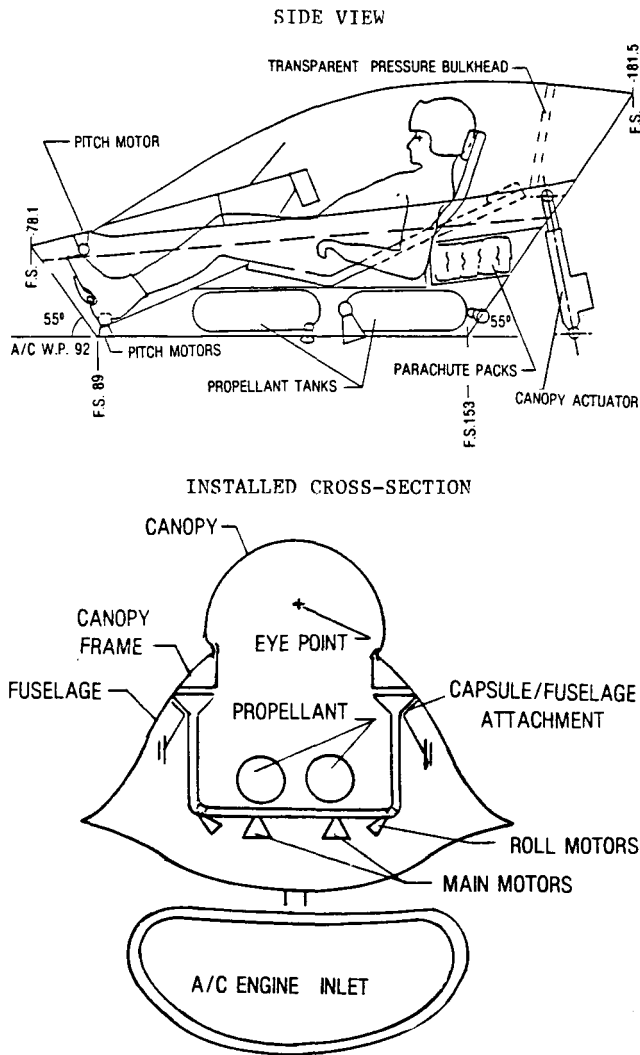


Figure 2. Canopy Escape Module

was recent data available on the F-16 and a prototype F-16 aircraft was at our disposal. The effort focused on providing safe escape capability up to a maximum 950 KEAS and the best combat acceleration (G) protection possible. The design really began with the G protection issue in order to define the pilot position. A capsule and escape subsystems were put around the pilot and fitted within the F-16 fuselage. This left a certain volume for the crew station which was less than optimum. The width of the F-16 at the crew station does not allow much room beside a reclined pilot. In the ideal case, the crew station requirements would have the opportunity to drive fuselage design. Following crew station design, aircraft to capsule interfaces were addressed and methods of ingress and egress in a CB environment proposed.

Reclined Seat & Minimum Weight Capsule

The approach to G protection was to recline the seat to reduce the column of blood between the head and heart. A 65 degree reclined seat was

designed that could provide protection up to 9 to 12 G without requiring the pilot to perform strenuous anti-G exercises while trying to fight the enemy. (See Figure 3 below.)

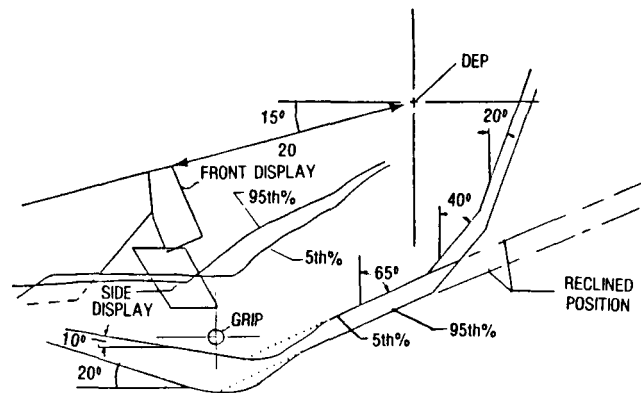


Figure 3. Reclined Seat Design with 5th and 95th Percentile USAF Male Pilot Outlines

Having a reclined seat helped reduce capsule weight by minimizing the cross-sectional area. This lowered the propulsion weight requirements by reducing capsule aerodynamic drag. The capsule was made insertable into the airframe to avoid having to accelerate aircraft fuselage structures during ejection. The body of the capsule was designed to be made of composite materials and molded around the minimum volume required for the crew station. The weight of the crew station was minimized by taking advantage of predicted technologies for the year 2010. The head-up display would be projected from the helmet, and voice control, eye tracking and artificial intelligence would be used to minimize the number of instruments and controls required. The front displays would be flat panel type to eliminate all CRTs from the crew station. Finally, all display generators, control units, radios, and other avionics would be outside the capsule with data transfer through fiber optics or coaxial cable. As a result the only weights required for the capsule are structure, escape systems, display and control input devices, and the crewmember. The resulting capsule weight to meet today's aircraft performance of 800 KEAS is just over 1200 lb and the associated weight penalty to the aircraft only increased by 27 percent (117 lb) over that estimated for an ejection seat based on the CREST technologies.

Escape System Design & Performance

This Canopy Escape Module (CEM) design was able to incorporate the same controller technologies being developed for the CREST demonstrator and added a highly controllable propulsion system. The propulsion system uses gel propellants which consist of thickened liquid oxidizer and fuel. These materials ignite instantly when they meet which eliminates the need for igniters and allows pulsing of the propellant for a very simple but effective throttling method. These propellants are highly efficient and can be used more efficiently compared to solid propellants which cannot

be started and stopped at will. The propulsion system was sized by estimating the amount of propellant required to bring the capsule to a stop from a maximum speed dive at sea level and comparing the results to the propellant required to recover the capsule at maximum velocity at Mach 3.5. It was estimated that a 950 KEAS capable capsule with 190 lb of propellant could recover the capsule with only one second remaining to impact in a 950 KEAS vertical dive. The high altitude case for a 950 KEAS capsule (42,000 ft) requires large amounts of propellant due to the high velocity of 3400 ft/s (compared to 1600 ft/s at sea level). It takes far longer to slow down from such a high speed, but the 190 lb of propellant was found to be adequate since the accelerations could be much lower. By having a gel propulsion system and adding an appropriate ground sensor, it was possible to add a retrorocket landing capability. The same propulsion system used for ejection would be reused to bring the capsule to a stop on the ground regardless of wind and slope conditions. This landing method can be achieved at a lower weight than airbags and can reduce the landing injury rate dramatically.

Crew Station Design & Considerations

The CEM crew station that resulted from reclining the pilot into a low-profile capsule led to some potential problems. The total surface area available for controls and instruments is only half of that for our current fighters. In addition, the reachable and visible area in flight is less than one-fourth of that currently available. Part of this severe limitation was caused by the F-16 fuselage and bubble canopy geometries. Some increase in side panel area can be expected in future fighter aircraft. However, the available space will still be very limited due to the reclined position of the pilot. This will require a tremendous change in crew station design and the overall pilot vehicle interface. To accomplish the necessary changes, the aircraft will have to have a pilot's associate to automate many of the functions currently being performed by the pilot. The best solution involves the use of the Super Cockpit as proposed by the Human Systems Division which would have the helmet project a complete computer enhanced world through the pilot's visor. This computer world would be overlaid on the real world to highlight threats, course information, targets, and assist the pilot's orientation. This super cockpit would also have an audio system that could cue the pilot in three dimensions by having threats and other things announce themselves to the pilot from specific directions. When combined with voice control, the pilot's ability to tell the aircraft systems what to do, these technologies would eliminate the current dependence on control surface area and a multitude of knobs and switches. Another effect of reclining the pilot is to make it more difficult to search the space around the aircraft for threats. This is most difficult in the rearward direction where much of the danger is coming from. Pilots currently whip their heads around to compare their attitude with the attitude of their enemy. Preventing them or making it difficult for them to do this would make this system impossible to achieve unless there is an

even better approach. Our efforts have led to a proposal that the pilot could take advantage of Super Cockpit system capabilities to allow the pilot to perform combat maneuvers while looking in the forward direction. The three dimensional audio capability would cue the pilot to the enemy's location and the helmet display would project a rearview mirror image at a location 180 degrees from the direction of the sound (and the enemy). This image would show the orientation of the enemy aircraft and would provide a consistent reference for the pilot to act on. The pilot could then maintain his own head and body attitude relative to his aircraft and would have easier access to the other information needed to maintain maximum aircraft performance. The pilot would stay in the best position for G protection and would be better able to maintain concentration and awareness during combat maneuvers.

Capsule/Aircraft Interfaces & CB Ingress/Egress

The fact that the capsule was designed to be insertable led to several benefits. The capsule and its volume became independent of other aircraft changes to a large extent. This way when fuselage weight increases in the nose, the capsule weight doesn't increase by default. Also, by being inserted into the fuselage, the capsule could be removed for maintenance of both the capsule and the fuselage components surrounding it. In order to accomplish this, some kind of latching mechanism would be required between the capsule and the airframe. There would also have to be a releasable interface between the two structures for power, data transfer, and environmental control system air. This removability of the capsule has the potential to provide a CB ingress and egress mechanism. The pilot could be left in the cockpit while it is removed and replaced by another pilot and cockpit. This would allow aircraft turnaround along with time for careful decontamination of returning capsules. Another approach for CB ingress and egress was proposed with less dependence on technology and support equipment. A plastic curtain could be deployed around the open crew station between the canopy and the canopy sill. This curtain would have a special zipper on it matching a zipper on a large impermeable suit. When these zippers are mated an airlock is created between the suit and the canopy (or shelter). This entire system might last several uses before being disposed of and replaced. (See Figure 4.)

CONCLUSIONS

Previous escape capsule experiences in the USAF have led to a reluctance to pursue such escape systems for future fighter aircraft. Our efforts have shown that emerging technologies will allow a capsule to be developed that offers the maximum escape system performance at a small increase in weight penalty to the aircraft. New technologies are available which allow safe escape over a larger portion of the aircraft flight envelope and can greatly reduce landing impact injuries. A concerted and integrated crew station and escape system design can lead to greatly improved pilot performance by providing increased G protection

ORIGINAL PAGE IS
OF POOR QUALITY

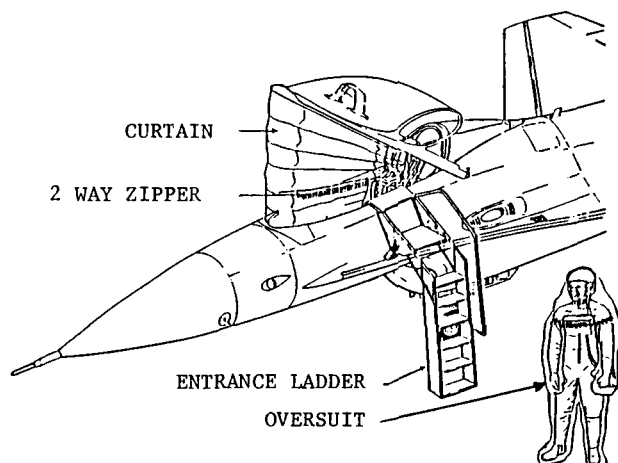


Figure 4. Proposed CB Protection System

and advanced crew station technologies. The CEM has been designed to achieve these improvements while reducing maintenance problems by having a removable capsule. The serious problem of ingress and egress in a CB environment can be overcome by using the removable capsule as a transfer means. Another approach has been proposed involving a plastic curtain around the open cockpit and a large plastic suit that can be zipped to the curtain forming an airlock.

BIBLIOGRAPHY

1. Schultz, E.R., A Canopy Escape System for Future Tactical Aircraft, Paper presented at 21st Annual Safe Symposium, Las Vegas, NV, November 1983.
2. Hubert, Lt J.A., Alexander, Lt K.K., Pickl, W.G., Concept Development of a Canopy Escape Module, AFWAL-TR-88-3049, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, OH, May 1988.
3. Crew Escape Technologies (CREST), Contract No. F33615-84-C-0518, Harry G. Armstrong Aerospace Medical Research Laboratory, Wright-Patterson AFB, OH, 1984 -.
4. Bull, J.O., et al., Compilation of Data on Crew Emergency Escape Systems, AFFDL-TR-66-150, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, OH, September 1966.
5. Akers, C.K., Ph.D., Pugliese, S.M., The Calspan Interchange Mechanism - A Novel Solution to Entrance/Egress Problems, Calspan Corporation, Buffalo, NY, February 1983.

CHEMICAL WARFARE PROTECTION FOR THE COCKPIT OF FUTURE AIRCRAFT

William G. Pickl
Air Force Wright Aeronautical Laboratories
Aircrew Protection Branch
Wright-Patterson AFB, Ohio, 45433-6553

ABSTRACT

Currently systems are being developed which will filter chemical and biological contaminants from crew station air. In order to maximize the benefits of these systems, a method of keeping the cockpit contaminant free during pilot ingress and egress is needed. One solution is to use a rectangular plastic curtain to seal the four edges of the canopy frame to the canopy sill. The curtain is stored in a tray which is recessed into the canopy sill and unfolds in accordion fashion as the canopy is raised. A two way zipper developed by Calspan could be used as an airlock between the pilot's oversuit and the cockpit. This system eliminates the pilot's need for heavy and restrictive CB gear because he would never be exposed to the chemical warfare environment.

INTRODUCTION

The Soviet's recent use of chemical warfare agents and toxins in Afghanistan shows their willingness to use chemical weapons to achieve their military goals. A chemical attack serves two purposes. The first is to cause direct casualties on enemy personnel. The second is to reduce the opposition's performance by forcing him to wear a restrictive Chemical and Biological (CB) protective ensemble while working in a contaminated environment.

Current aircraft have no means of preventing the contamination of the cockpit interior when the airfield has been chemically attacked. Because the cockpit is "dirty" the pilot must wear a heavy protective suit throughout his mission. The current CB ensemble consists of: a MBU-13/P chemical-biological-oxygen mask (CBO), a butyl rubber hood, a CRU-80/P charcoal filter pack, an active charcoal impregnated undergarment, and butyl rubber glove inserts. In order to protect the pilot from liquid contaminants, a plastic overcape and overboots are worn by the pilot until he reaches the aircraft. The CB ensemble reduces pilot performance by decreasing head mobility and peripheral vision, and by increasing heat build-up underneath the CBO and the rubber hood. Although the current CB protection system defeats the first purpose of a chemical attack, it falls victim to the second. This lead to our development of a CB protection system concept which protects the pilot from CB agents in a manner which minimizes the negative impact on pilot performance (see figure 1).

CHEMICAL BIOLOGICAL PROTECTION SYSTEM

The proposed CB protection system consists of an ingress/egress system and a Cockpit Atmosphere Protection System (CAPS). The ingress/egress system would keep the pilot from being exposed to contaminants during his transfer from the protection shelter to the aircraft. The CAPS would maintain a contaminant free environment inside the cockpit.

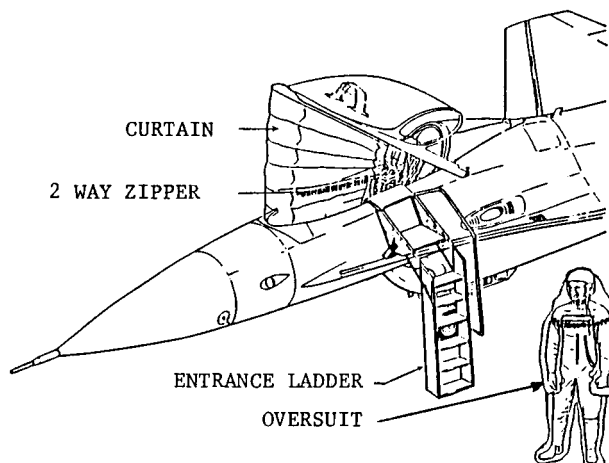


Figure 1. PROPOSED CB PROTECTION SYSTEM

INGRESS PROCEDURE

The new ingress procedure would begin with the pilot entering an impermeable oversuit using a two-way zipper (such as the "Supertab" developed by Calspan for the Army) to form an airlock between the protective shelter and the oversuit. The pilot would then travel out to his aircraft protected by the oversuit. He would then ingress the aircraft by using the zipper to form an airlock between the pilot's oversuit and CAPS. The pilot would ingress and egress the cockpit through a 3' 6" long two-way zipper located on the side of the curtain (see figure 2). Once inside, the pilot would zip the curtain closed and the oversuit would fall off. To egress the cockpit, the pilot would first zip his spare oversuit onto the inside of the curtain. He would then turn the suit inside out, pushing it through the opening in the

ORIGINAL PAGE IS OF POOR QUALITY

curtain so that it would then be on the outside of the curtain. He would then reverse the ingress procedure and return to the shelter. The use of overpressure insures that no contaminants enter the cockpit through the zipper mechanism. The cockpit air used to provide the overpressure could be cleansed by a catalytic filtration system such as the system being developed by the Environmental Control group of the Air Force Wright Aeronautical Laboratories. The oversuit and the CAPS would protect the pilot from the contaminated environment. This would allow him to fly his mission free from the hot and cumbersome CB ensemble.

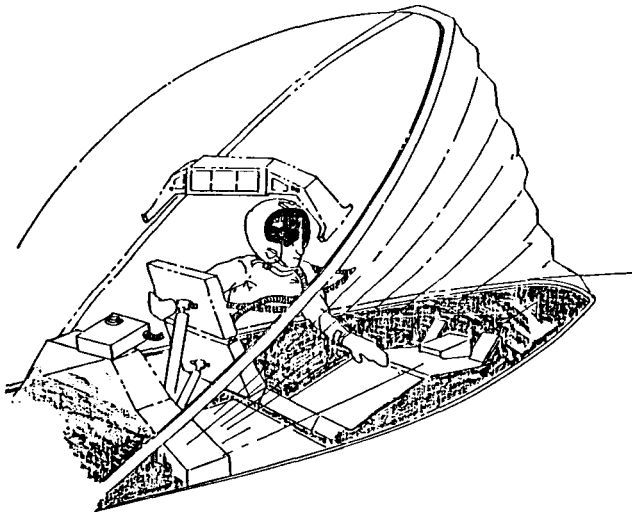


Figure 2. PILOT TRANSFER FROM SUIT TO COCKPIT

OVERSUIT

The oversuit would be designed to protect the pilot from liquid and vapor contaminants during transfer to and from the aircraft. It would be constructed from a lightweight (approximately 6 oz per square yard), flexible, and tough material, which has good CB barrier properties (one possible material is a Saranex barrier film bonded to a Tyvek support material). The suit would be disposable and designed so that one size would fit the 1st through 99th percentile U.S. Air Force pilots. The suit would be large enough so that the pilot could crawl into and out of it easily using a 3' 6" opening across the chest, which would be sealed by one half of the two-way zipper. The pilot would be wearing an advanced flight suit which would provide filtered air during his transfer to the aircraft and some CB protection in case of emergency egress or a leak in the CB protection system.

COCKPIT ATMOSPHERE PROTECTION SYSTEM

The CAPS consists of a RECTANGULAR CURTAIN, a STORAGE TRAY, a STORAGE TRAY LID, INFLATABLE RUBBER SEALS, and a RETRACTION SYSTEM each of which are described below.

The RECTANGULAR CURTAIN (see figure 3) is designed to protect the cockpit from CB agents when the canopy is raised by sealing the edges of the canopy to the canopy sill. The curtain could consist of a thin sandwich (approximately .006 inches 6 oz per square yard) of a nylon polymer film laminated to a support substrate. The curtain unfolds and folds in accordion fashion as

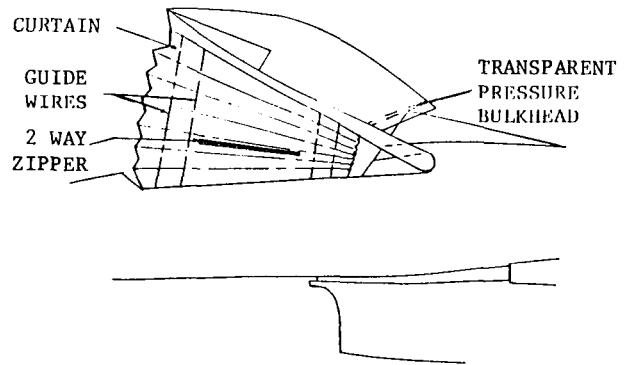


Figure 3. SIDE VIEW OF CAPS

the canopy is raised and lowered. The curtain dimensions were developed using the F-16 cockpit as a baseline. Assuming a 3" limit in folded width the curtain would require 48 panels to seal the cockpit opening. Because the curtain is shorter in the back, the width of the folds would only be 1". The width of the folds along the sides tapers from 3" at the front to 1" at the rear. The curtain is attached at the bottom to the storage tray and at the top to the storage tray lid. To prevent the curtain from billowing out from overpressure, plastic stiffeners would be attached to the inside edge of each curtain fold (see figure 4). The stiffeners would be constructed of a material which was 2" wide and .04" thick. The material would be capable of being creased at the center and folded 180 degrees. The combined weight of the curtain and stiffeners would be approximately 15 pounds.

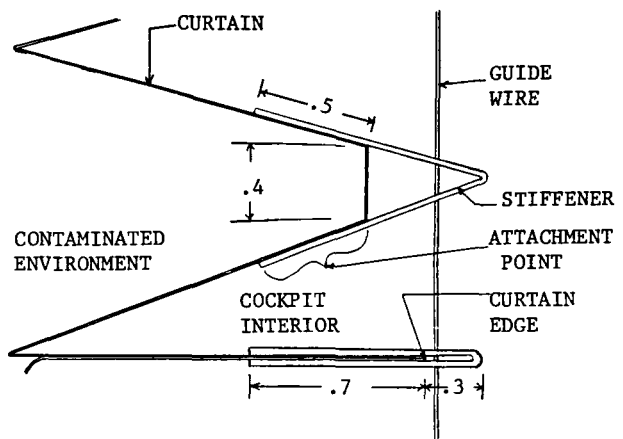


Figure 4. CROSS CUT VIEW OF CURTAIN
(Dimensions are in inches)

The STORAGE TRAY (see figure 5) could be formed of a single molded plastic tray which would be impermeable to chemical agents and weigh about 4 pounds. The storage tray would be inserted into a recessed groove in the canopy sill. The width of the bin would be 4" at the front of the sill and 2" at the back and would vary linearly from 4" to 2" along the sides. The depth of the bin could be 3.5" and would depend on curtain thickness. The seal insert would attach the storage tray to the bottom of the recessed groove in the canopy sill. The small indentation at the bottom of the storage tray is the guide wire channel and along its length are sockets which would hold the guide wire rollers in place. To the right of the guide wire track is the curtain attachment strip.

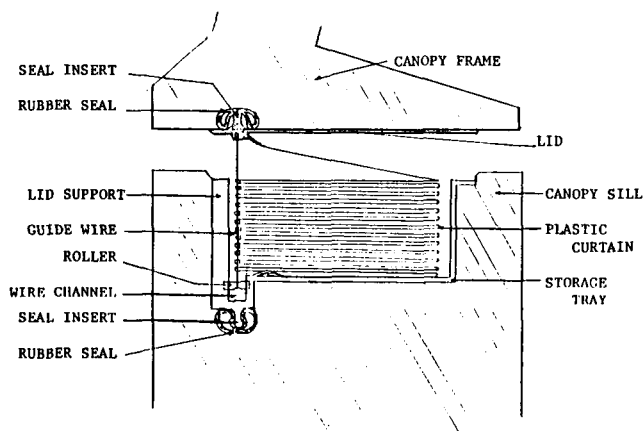


Figure 5. CROSS SECTION OF CAPS

The STORAGE TRAY LID could be formed of a single piece of 1/16" thick hard molded plastic, which would also be impermeable to chemical agents and would weigh about 1 pound. When the CAPS is not in use, the primary purpose of this piece is to serve as a lid to the curtain storage tray and is secured to the storage tray by the tension in the guide wires. When CAPS is in use, the storage tray lid serves as the means by which the curtain and guide wires are connected to the canopy.

The INFLATABLE RUBBER SEALS could be made of a butyl rubber and would weigh approximately 2 pounds. The seals could be rounded on their outer surface and have two hollow channels, one on each side. When the channels are pressurized by a CO2 cartridge which is attached to each seal, the seal squeezes the seal insert. The beaded shape at the end of the seal insert prevents it from pulling loose. One seal would be placed in the storage tray recess of the canopy sill and the other would run around the canopy edges.

The RETRACTION SYSTEM would consist of guide wires, guide wire rollers, and a tensioning mechanism. The guide wires could be made of nylon and would be capable of withstanding 50

pounds of tensile force. Two guide wires would be located on each side of the curtain. The wires would be anchored to the lid on one end, pass through grommets on the stiffeners, go around the rollers, pass through the wire channel, and attach at their other end to the tensioning system. The purpose of the guide wires is to ensure that the curtain folds properly into the storage tray. They would also support the curtain and prevent excessive billowing due to over-pressure or wind. The tension system could consist of either an electric motor and take up reels or some type of spring tension system. The total weight of the tension system is estimated to be 10 pounds.

OPERATION

While the aircraft is in a safe environment, the inflatable rubber seal in the canopy frame would be left deflated. This would leave the storage tray lid attached to the storage tray, allowing normal ingress and egress. Once the aircraft has been exposed to contaminants the inflatable rubber seal inside the canopy frame would be pressurized by its CO2 cartridge before the canopy is raised. The CO2 cartridge can be activated from either the inside or the outside of the cockpit. The storage tray lid is now firmly attached to the canopy frame. As the canopy is raised the lid pulls the plastic curtain out of the storage tray and the curtain stiffeners slide up the guide wires. As the canopy is lowered the guide wires insure that the curtain folds into the storage tray properly. The seals would remain pressurized until there is time to decontaminate the aircraft and replace the CAPS.

CONCLUSION

The CAPS would be a light weight (approximately 35 pounds) device which would prevent contaminants from entering the cockpit. The CAPS has the advantage of remaining passive until CB agents are detected, when it can be activated by the push of a button. CAPS can be removed and replaced quickly and easily when the aircraft has been decontaminated. CAPS makes use of a double zipper mechanism which, when combined with an impermeable oversuit, allows the pilot to transfer safely from a protective shelter to the cockpit. This means the pilot could transfer to his aircraft without ever being exposed to the CB environment which would defeat the first purpose of a chemical attack. The CAPS would allow the pilot to maintain the same performance level he had before the airfield was contaminated, defeating the second purpose of a chemical attack.

REFERENCES

1. "Soviet Chemical Weapons Threat," DST-1620F-051-85, Department of Defense Intelligence, 1985.

2. Merrifield, John, "USAF Crews Increasing Proficiency In Chemical Defense Clothing," Aviation Week and Space Technology, July 28, 1986, 73-78.
3. Hubert, James Capt, Alexander, Kristen Capt, and Pickl, William, "Concept Development Of A Canopy Escape Module," AFWAL-TR-88-3049, Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson Air Force Base Ohio, May, 1988.
4. Thompson, Al, and Henneman, Teresa, "Technology Impacts Of Advanced Technology Crew Protection (ATCP)," AFWAL-TR-85-3074, Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson Air Force Base Ohio, December 1985.
5. Pilie, Roland, et al., "Interchange Mechanism For Multiple Fasteners," Patent number 4,485,534, United States Patent, Dec. 4, 1984.
6. Pilie, Roland, et al., "Entrance And Egress System For Protective Shelters And Garments," Patent number 4,485,489, United States Patent, Dec. 4, 1984.

OPTIMAL SOLUTIONS, FOR COMPLEX DESIGN PROBLEMS: USING
ISOPERFORMANCE SOFTWARE FOR HUMAN FACTORS TRADE OFFS

Robert S. Kennedy*, Marshall B. Jones**,
Dennis R. Baltzley*
*Essex Corporation
1040 Woodcock Road, Suite 227
Orlando, Florida 32803

**Milton S. Hershey Medical Center
Pennsylvania State University
Hershey, Pennsylvania 17033

ABSTRACT

A major application of isoperformance is as a trade-off methodology of the three major drivers of system design; equipment, training variables, and user characteristics. The flexibility of isoperformance allows each of these three components to be nearly any rational variation. For example, aptitude may be military Armed Forces Qualification Testing (AFQT) categories, cutoff scores within a selection procedure, or simply dichotomizing high and low scorers (pass/fail). Equipment may be new versus old, "smart" versus "dumb", high versus low resolution, etc. Training may be short versus long or varieties of media types (lecture versus CAI/CBI versus self-paced workbooks).

In its final computerized form isoperformance lets the user set an operational level of performance (e.g., a jet pilot in a simulated emergency must take prescribed corrective action and clear the plane in several seconds, pilot astronauts will check out all shuttle flight systems within 30 minutes, or Mission Specialists must handle successfully a required number of job elements). At this point the computer program guides the user through any requested trade-offs of the three components while maintaining the specified operational level of performance through "isoperformance curves." A demonstration of the computer program is currently available.

INTRODUCTION

Since the 1950's applied behavioral scientists working in the fields of systems, training, and selection have remained largely independent from each other. Within most organizations the various policy management guides and functional mission statements reinforce this separation. Historically, in systems research work, human factors practitioners have been taught that their role is to gather these human input/output data (transfer functions) and determine how they interact with their equipment (or physical and

environmental stimuli). These data would then be used to generate standards and specifications which could then be used by design engineers to improve systems performance. Human factors experts also believed that design engineers were eagerly awaiting these data to incorporate into new systems which would permit efficient allocation of functions between man and machines (Fitts, 1951; Taylor, 1963). This goal was naive; attention must be given to techniques whose goals are to improve decision making in systems research by employing as a strategy the notion of "trade-off technology."

The isoperformance approach (Jones, Kennedy, Kuntz, & Baltzley, 1987) is based on the premise that differing combinations (i.e. trade-offs) of individual differences, training, and equipment variables can lead to the same desired outcome in total operational systems performance. It is called isoperformance (iso meaning same) and is a conceptual approach to systems research in human engineering. The key ingredient of isoperformance is to invert the question of operational performance enhancement by setting a desired level of performance and derive how it can be attained by different combinations of personnel, training, and equipment. The goal is that once these combinations have been determined, choices among them can be made in terms of maximum feasibilities or minimum costs. The program takes into account technology advancement and systems, personnel, and training research. It leaves an audit trail of the decision process.

Development of the Isoperformance Concept.

The isoperformance concept originally surfaced out of our scientific human engineering studies and experiences on military flight simulators. The simulator experiments sought to identify which equipment features best promoted acquisition of flying skills (Lintern, Nelson, Sheppard, Westra, & Kennedy, 1981) and followed the holistic design philosophy of Simon (1976). This approach reports the size of main effects of "equipment features" like display resolution, luminance and contrast or scene

content, and permits comparison of these effects (in terms of size) with reliable individual differences like visual contrast sensitivity, dark focus, or video game performance, as well as training improvements. The data from these studies were generally reported in terms of percent of variance accounted for, for each variable, and a meta analysis of the relative contributions has been made over all the studies (Jones, Kennedy, Baltzley, & Westra, 1988) over a nine-year period of our association with the project. The present authors concluded that in these studies, when comparing aptitude, training, and equipment, reliable individual differences (aptitude) explained substantially more (usually twice as much) of the variance in the task performance than either practice or equipment. From this relationship has stemmed the isoperformance concept of trading off the three performance predictors where it appeared that individual differences (reliable differences, not error) played a large part.

In earlier conceptualizations an Omega Squared (Hays, 1977) meta analysis was attempted for a large portion of the published human factors literature in order to identify the relative contributions of individual differences versus training versus equipment features. In this work, over 10,000 citations from the body of literature in the field produced less than 0.1% which possessed and reported data in sufficient detail in order to be able to perform an adequate analysis of effect size (Jones, Kennedy, Turnage, Kuntz, & Jones, 1986). These findings presented a sobering commentary on the state of the existing literature for grounding the development of trade-off methodologies on empirical findings.

However, there are several other alternatives. Expert judgments, based on knowledge of the technical literature, can be heavily constrained to make inferences in the form of estimates (Jones, Kennedy, Kuntz, & Baltzley, 1987). Alternatively, formal experiments can be carried out and implemented under an innovative technical framework such as we (Kennedy, Jones, & Baltzley, 1988) have showed where a considerable amount of the explainable variance in task performance remained after blocking out the results according to dictates of the isoperformance methodology. Other methods also exist when empirical data is unavailable.

Structural Elements in Isoperformance. There are three main elements in the isoperformance methodology. These include individual differences, training, and equipment variations. Several synonyms are used in this report for these three dimensions.

a. Individual Differences. These differences include all of the many identifiable characteristics of people from

sensory sensitivities, strength and anthropometric variables to mental capabilities and motor skills. The military, for example, employs the multidimensional Armed Forces Qualifications Test (AFQT) where anyone classified below Category 4 is not accepted (Maier & Grafton, 1980). Nevertheless, even with these restrictions in range (Sims & Hiatt, 1981), individual differences among military personnel are great. For example, in naval aviation, stringent visual examinations are used for acceptance, yet the distance at which one pilot customarily can detect opponent aircraft is sometimes 50-70% better than another, resulting in 2-3 mile advantages in early detection. Moreover, some pilots who are better at visual detection can even "outsee" the poorer ones when the latter use telescopes (Jones, 1981, personal communication). In this example, if equipment factors were evaluated to determine effects on performance in terms of the amount of accountable variance, one could not adequately assess the question without taking into account the differing visual and perceptual capabilities of the individual pilots. Cognitive and other mental capabilities also show wide variation (cf., Schoenfeldt, 1982, for a review). These relations are similarly available in industry and business although perhaps not as well documented.

b. Training. Training or practice can also be viewed under several different rubrics such as the number and length of trials, instructional systems (e.g., lecture, on-the-job, text), simulator vs. embedded training, or the type of practice regimen (massed versus distributed). In the discussions which follow, when we employ these and other denotations to report the specific outcomes, it is always our intention to connote the more general notion of the broader class of the dimension.

Specifically, a recent review of the lawful relationships from the scientific literature related to training has been completed (Lane, 1986). The sheer magnitude of the information in the report defies simple explanation. For example, learning curves vary in their shape. Tasks that are primarily conceptual may show plateaus or large gains with short amounts of practice. However, skill acquisition and procedural tasks generally show the "traditional learning curve." The shape of the learning function is such that the most rapid amount of training effect occurs initially and the best description of the overall relationship is that log performance (or practice) is a linear function of log practice (Newell & Rosenbloom, 1981). Thus, ranges of improvement in performance during formal training can be an order of magnitude of improvement for each epoch of time spent in training (cf. Hagman & Rose, 1983; Lane, 1986; Schendel, Shields, & Katz, 1978). Improvements of as much as 500% are not unusual. Such a

range of improvements can temper any expected change due to equipment factors and, because of their size, must be included in planning for technological design of the workplace.

The problem outlined above is not one which will lessen with time but rather the converse. It is believed that the problem of function allocation becomes more critical with the growing complexity and sophistication of machine systems. Considering the survey of the literature (Jones et al., 1986) it is believed a systematic methodology, such as isoperformance, can be provided to account for man/machine interface problems and present decision aids to create trade-off alternatives from the human side of the combination with no loss of operational proficiency.

c. Equipment. By equipment comparisons we may mean features that can be varied on a single piece of hardware (e.g., brightness, resolution or contrast) or several disparate engineering options (e.g., artificial intelligence versus unaided displays) or different software modifications (e.g., rate aiding, predictor display). Equipment is also a term that can encompass many of the new workplace technologies including office automation systems and computerized manufacturing systems. Under certain circumstances "equipment" could mean two models or versions of a system or it could be simulator versus actual aircraft.

The Isoperformance Methodology. Cost-effective methods may proceed in either of two general ways. The more familiar is to fix costs and maximize effectiveness. One gets, as the popular phrase puts it, "the biggest bang for the buck." The alternate procedure is to fix effectiveness and minimize health, safety, personnel, training, equipment, and manpower costs -- to get "the same bang in the least costly and most expeditious way." This latter approach leads naturally to trade-offs among the cost factors and is the approach taken by isoperformance methodology (Jones et al., 1987).

The heart of this methodology is the isoperformance curve. With respect to aptitude levels and training times such a curve looks like the one given in Figure 1. The Y-axis is aptitude measured, for example, by cut-off scores. For this example let's suppose there were five categories within which incumbents or applicants could fall (two high, two low, and a middle category) on a particular aptitude test. The X-axis is training time in weeks. The job might be that of a computer operator. The curve drawn is for 80% proficient. That is, any point on the curve (any of the indicated combinations of aptitude level and training time) will produce personnel 80% of whom are proficient at the job. Thus, if one

has high-aptitude applicants (for example, in the highest cut-off categories) 80% proficient can be reached in roughly eight weeks. With lower aptitude people more training time is needed and for some aptitude levels (The lowest cut-off scores) no amount of training time up to the maximum considered will suffice to produce computer operators 80% of whom are proficient.

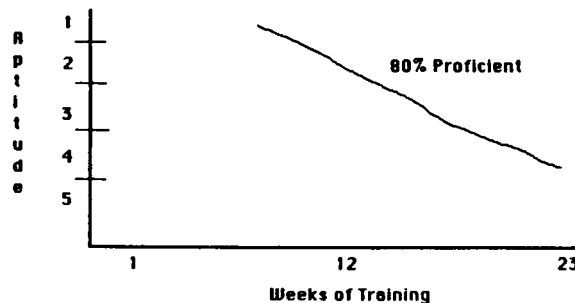


Figure 1. An isoperformance curve for 80% proficient.

Isoperformance curves come in families. A separate and distinct isoperformance curve exists for every level of performance that one specifies. Thus, if one were to specify 50% proficient, for example, one would get a different curve than the one that appears in Figure 1. Note that the second curve (Figure 2) lies to the left and down from the first curve presented. It takes less time to train the same people to the lower level of performance or, in the alternative, for the same amount of training time the lower level of proficiency can be attained with lower aptitude personnel.

A pair of curves quite similar to the pair in Figure 2 can be obtained in a quite different way. Suppose one were to automate part of the computer operator's job by providing him/her, perhaps, with more advanced computer equipment that was itself easy to use (which is done with some regularity). With the new equipment the job becomes considerably simpler so that the same objective results can now be achieved by lower aptitude people or with less training time. The situation is depicted in Figure 3. Again there are two curves, but this time the two curves correspond to two equipment variations and both represent the same level of performance. Any point on either curve suffices to produce personnel 80% of whom are proficient. Using the new equipment the same people can be trained to the same level of performance (80% proficient) in less time. Or, for a given amount of training time, the same level of performance can be achieved with lower aptitude personnel.

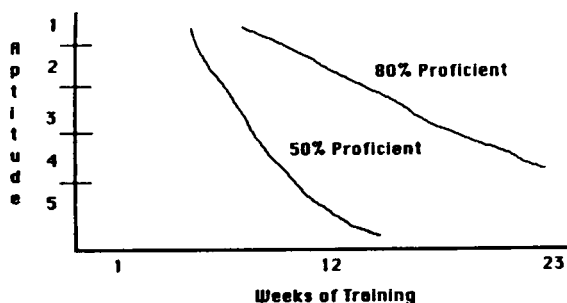


Figure 2. Two isoperformance curves, one for 80% and the other for 50% proficient.

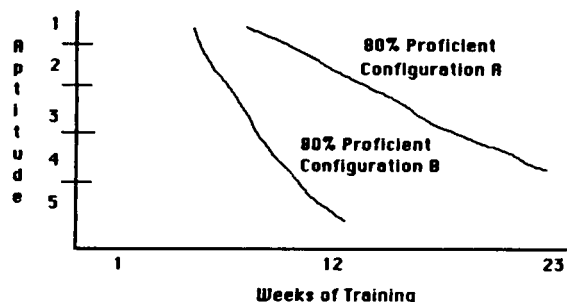


Figure 3. Two isoperformance curves, one for each of two equipment configurations, but both for the same job and the same level of performance.

Isoperformance curves must be evaluated before any conclusion can be reached. Any point on either of the two curves in Figure 3 will produce 80% proficient personnel -- but which point is best? To answer this question one invokes other cost considerations. Category 1 and 2 people may be in such demand for other jobs that they must be regarded as unavailable. Training times in excess of 12 weeks may be excessively expensive. Figure 4 re-presents Figure 3 marked to reflect these two considerations. Since category 1 and 2 personnel are excluded by reason of unavailability, and category 3 personnel (or lower) require more than 12 weeks to reach 80% proficient using the original equipment, there is no solution to be obtained using equipment configuration A. The alternative equipment, however, does provide a range of solutions. Any point on the lower curve between the horizontal and vertical bars would be acceptable insofar as personnel availability and training costs are concerned. They might not be equivalent, however, on other counts. It might be, for example, that training schools for computer operators must extend at least eight weeks, shorter lengths of time being impractical for scheduling reasons. The solution would then have been narrowed to the second equipment configuration (B), category 3 and 4 personnel, and a training time between eight and twelve weeks.

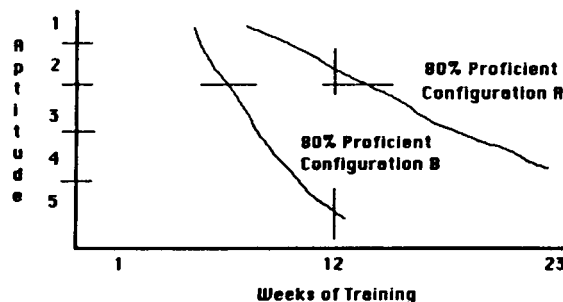


Figure 4. Figure 3 marked to indicate that category 1 and 2 people are not available and that training times in excess of 12 weeks are too expensive.

Isoperformance is a very powerful aid to decision making and becomes more indispensable as the organization becomes more complex. Thus far, it has been explored in the environment of military systems but the implications are much broader. Moreover, because of the greater requirement to utilize and train available manpower, much of the potential power and flexibility of the model can be taken advantage of in civilian applications as outlined in a later section. The final computer program may be used not only by system design specialists, but also by executive decision makers, human resource and training specialists, as well as human factors engineers doing strategy planning.

In addition to related work like the Army's MANPRINT, the Navy's HARDMAN, and the Air Force program RAMPARTS, some of which have been referred to above, we offer as background some of the work on isoperformance. Isoperformance, in addition to referring to the computer program which is developed for the Air Force, is also a philosophy or conceptual model which structures how one addresses man-machine interactions. All of these trade-off technologies therefore have considerable features in common. What distinguishes the isoperformance work to some extent is the series of interactive software programs which are in the process of completion. The next development in that Air Force work is IsoCore. A preliminary version of this planned computer program will be available in FY88 and is described in some detail below.

IsoCore. Suppose we are given a known or designed piece of equipment and a definition of what "proficiency" means for a task to be performed using this equipment. Suppose further that if no data is available the program's user is able to estimate certain training outcomes for different categories of personnel (which outcomes will be specified shortly). Isoperformance is intended to achieve the following aims:

ORIGINAL PAGE IS OF POOR QUALITY

- It forces the user to make estimates of training outcomes for different personnel categories;
- It provides checks on the internal consistency and logical coherence of these estimates;
- It provides checks on how well the estimates conform to known regularities from research in systems, human engineering, personnel, and training;
- It informs the user as to the results of these checks, together with information about what can be done to make the estimates consistent or bring them into closer conformity with known regularities and facts;
- It outputs "isoperformance curves," that is, curves in a space defined by aptitude and training time all points of which are estimated to produce the same proportion of proficient personnel;
- It leaves a hard-copy audit trail of all estimates, feedback, and outputted isoperformance curves.

The isoperformance core subprogram is being written in four phases: specification, input, verification, and output. These phases will be discussed in the order given.

Specification. The first phase of the core subprogram requires the user, in effect, to state the problem. The user is asked to specify:

- the system under study
- the task to be performed
- what is meant by "proficient" performance
- the personnel population to be considered
- the aptitude dimension to be used as predictor
- how that dimension is to be divided into ranges or "aptitude categories"
- the training program
- the maximum amount of training time to be considered.

With two exceptions, the main purpose of these specifications is to provide a basis for checking the user's input estimates against known training outcomes and predictive validities for similar tasks and personnel categories.

The isoperformance core subprogram provides for a single predictor dimension. This one dimension does not, however, have to be unitary in a factor-analytic or any other sense. In the usual case it will be a maximally predictive linear composite of aptitude variations related to performance on the task under consideration. The specified personnel

population is divided into ranges by cut-off points on the one predictor dimension. The cut-off points themselves are not important. What matters is the proportion of the subject population that falls into each category. The maximum amount of training time to be considered must be specified because, otherwise, training time becomes indeterminate. The instructions to the user are to specify the maximal amount of time that could, taking cost considerations and other demands on instructional personnel and facilities into account, be considered feasible for training people to perform the specified task.

Input. For each aptitude category the user is required to make either two or three estimates. The first of these estimates is the amount of training time necessary for 5% of category one personnel to become proficient; the second is the proportion of persons in the category who will be proficient given the maximum amount of training time considered feasible; and the third estimate is the amount of training time necessary to make 50% of category one personnel proficient. Plainly, this third estimate is needed only if the second is greater than 50%. These two or three points define a rough skill acquisition curve. If the second estimate is greater than 50%, however, the negative acceleration of the mean performance curve is intensified once the median category member becomes proficient. At that point the distributional effect also becomes negatively accelerated making for a relatively sharp "turn" at 50% proficient.

Verification. The third phase of the ISOPERFORMANCE core subprogram checks to make sure that input estimates are "reasonable." Doing so involves three kinds of checks: formal, general, and specific. A formal check is analytic, that is, a matter of logical necessity. The estimates, for example, should increase or remain the same with decreasing aptitude category. The second kind of check (general) is for conformity with known regularities from personnel and training research. The third kind of check (specific) involves comparing the input estimates and extracted correlation coefficients with known training outcomes and predictive validities for similar tasks and personnel categories.

Output. Output consists of easily understood isoperformance curves on a graph of aptitude on the ordinate and training time on the abscissa. This background outlines the extensive groundwork which has been done formulating the isoperformance concept for military applications. This work will also serve as the basis for which to empirically ground the transition to civilian industry as a useful decision making tool for organizational and personnel cost assessment (in time and dollars).

COMMERCIAL APPLICATIONS

Isoperformance methodology has broad application in systems research for government and private industry. Five major areas of application are: (a) as a management decision aid for human factors engineering design; (b) as an adjunct to aid executives in organizing manpower, personnel, and training (MPT) applications, particularly where "what if" questions need to be answered and where an audit trail of the solution adopted is useful; (c) as a formal system for conducting trade-offs where cost analyses are conducted for existing systems; (d) as a way for industry to meet the functional specifications and requirements in an RFP; and (e) as a way for industry to be responsive to governmental contracts, especially those adhering to new guidelines on using NDI procurement strategies. A brief example is provided for each of these areas to demonstrate the utility of isoperformance methodology.

Isoperformance kinds of estimates are already required in the military in the form of MANPRINT analyses. The data available from the MANPRINT requirements for systems will work well as data for explicit trade-offs in isoperformance analyses. From these current data "ground-up" HFE design work could be pursued with maximally efficient systems as a result.

Within the MPT arena isoperformance methodology permits trade-offs for each component and provides immediate feedback for forecasting efficiency and selection/placement. The current IsoDemo program developed for the Air Force (Jones & Jones, & Essex Corporation, 1987) provides a constrained example of using the isoperformance methodology. At the end of the program the manager can tell what the lowest aptitude category is within the training time allotted and equipment constraints available. Conversely, he/she can also find the minimum training time necessary if the very best people were available which is seldom the case in the military; however, this will differ for private industry selection.

The third major area of isoperformance has the broadest application. This area is using isoperformance methodology for existing systems. Isoperformance can be used to evaluate and suggest improvements in any system where there is a man/machine interaction or the various costs of the different parts can be compared. This is especially useful with emerging technologies. In private industry technology changes weekly, isoperformance allows the decision-maker to evaluate each potential upgrade or changeover from a complete systems viewpoint and to make better informed choices from a organizational cost/benefit perspective.

Finally, industry may use isoperformance methodology to meet the functional specifications and requirements in an RFP or simply to be responsive to a customer's needs. Suppose the government or another large organization calls for updating or replacing an in-place piece of equipment. A company may propose to modify the system by upgrading it to make it "state-of-the-art," or it can trade off the complexity through longer training time or selection of higher aptitude personnel. The company may propose to replace the equipment with a less complex system with no development cost associated. In this way the company cannot only lower the unit cost but could provide isoperformance verification for shorter training time and broader use of the labor pool. This would result in substantial lowering of total system costs in training, personnel, and support. The benefits are obvious; the company may elect to pursue a technological advantage or an overall cost advantage. Both are defensible and may be suggested to a manager for overall preference. If the system is a simulator, state-of-the-art may be required. If it is a vehicle, an overall cost approach may be chosen. The Army, for example, adapted the Chevy Blazer to meet their light truck requirements.

As a computerized decision aid in design, the isoperformance program may be used to trade-off the aptitude, equipment, and training dimensions which are known or can be estimated for a prospective system. In this way overall utility as well as cost/benefit considerations may be assessed. For example, in a new weapons system, the projected manpower of the target service as well as the allowable minimum and maximum training times may be reasonably estimated. This will form a "window" within which the equipment (man/machine interface) must stay. Many questions about which elements to emphasize can be answered almost immediately by framing the question within the context of the isoperformance model.

Additionally, as a fifth point, the isoperformance approach is well suited for application of the recent policy mandating the use of Non-Developmental Items (NDI) in the military acquisition process. This NDI procurement plan is a direct result of the President's Council on Defense Acquisition, The Packard Commission. Governmental agencies are required to evaluate the ability of an "off-the-shelf" item for satisfying their functional needs. An NDI may be entirely off-the-shelf needing no development, or the item may require a dedicated R&D effort by the contractor to modify the item for current governmental needs. A major principle in NDI acquisition is that less than full compliance with a programs performance objectives is insufficient reason not to use NDI. In other words, if an NDI does not meet all specifications and requirements set forth in the Request for Proposal (RFP), it is not

ORIGINAL PAGE IS OF POOR QUALITY

disqualified; cost/benefit trade-offs can be made. Here lies the isoperformance strong point. Industry which deals in government contracting may invoke these NDI concerns and use isoperformance to trade off any weaknesses in their "off-the-shelf" products to maintain a more flexible position in competition.

REFERENCES

- Fitts, P. M. (Ed.). (1951). Human engineering for an effective air investigation and traffic control system. Washington, DC: National Research Council.
- Hagman, J. D., & Rose, A. M. (1983). Retention of military tasks: A review. Human Factors, 25, 199-213.
- Hays, W. (1977). Statistics for psychologists. New York, NY: Holt, Rinehart, & Winston.
- Jones & Jones, & Essex Corporation. (1987). A brief introduction to Isoperformance methodology [Computer program]. Orlando, FL: Essex Corporation. (Armstrong Aerospace Medical Research Lab. Contract No. F33615-86-C-0553)
- Jones, M. B., Kennedy, R. S., Baltzley, D. R., & Westra, D. P. (1988, in preparation). Omega square analyses and reevaluation of a meta-analysis for equipment features and individual differences in an aircraft simulator. Orlando, FL: Essex Corporation.
- Jones, M. B., Kennedy, R. S., Kuntz, L. A., & Baltzley, D. R. (1987). Isoperformance: Trading off selection, training, and equipment variations to maintain the same level of systems performance. Proceedings of the 31st Annual Meeting of the Human Factors Society (p. 634-631). Santa Monica, CA: Human Factors Society.
- Jones, M. B., Kennedy, R. S., Turnage, J. J., Kuntz, L. A., & Jones, S. A. (1986). Meta-analysis of human factors engineering studies comparing individual differences, practice effects, and equipment design variations (Final Rep. No. F33615-85-C-0539). Wright-Patterson AFB, OH: Department of the Air Force, Air Force Systems Command, Aeronautical Systems Divisions. (NTIS No. AD A168 790)
- Kennedy, R. S., Jones, M. B., & Baltzley, D. R. (1988). Empirical demonstration of Isoperformance methodology preparatory to development of an interactive expert computerized decision aid (Final Report for ARI). Washington: Army Research Institute.
- Lane, N. E. (1986). Skill acquisition curves: An analysis of applicability to the time course and sequencing of military training (Final Rep., Contract No. 5540, Institute for Defense Analysis). Orlando, FL: Essex Corporation.
- Lintern, G., Nelson, B. E., Sheppard, D. J., Westra, D. P., & Kennedy, R. S. (1981). Visual Technology Research Simulator (VTRS) human performance research. Phase III (NAVTRAEQUIPCEN 78-C-0060-11). Westlake Village, CA: Canyon Research Group, Inc.
- Maier, M. H., & Grafton, F. C. (1980). Renorming ASVAB 6/7 at Armed Forces examining and entrance stations (Technical Memorandum 80-1). Washington, DC: Office of the Assistant Secretary of Defense.
- Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), Cognitive skills and their acquisition. Hillsdale, NJ: Erlbaum.
- Schendel, J. D., Shields, J. L., & Katz, M. S. (1978). Retention of motor skills: Review (Technical Paper 313). Alexandria, VA: Army Research Institute.
- Schoenfeldt, L. F. (1982). Intra-individual variation and human performance. In M. D. Dunnette and E. A. Fleishman (Eds.), Human performance and productivity: Human capability assessment. (Vol. 1, pp. 107-135). Hillsdale, NJ: Lawrence Erlbaum.
- Simon, C. W. (1976). Analysis of human factors research engineering experiments (TR-CWS-02-76). Westlake Village, CA: Canyon Research Group.
- Sims, W., & Hiatt, C. M. (1981). Validation of the Armed Services Vocational Aptitude Battery (ASVAB) forms 6 and 7 with applications to ASVAB forms 8, 9, and 10. Alexandria, VA: Center for Naval Analyses. (NTIS No. AD A110 025)
- Taylor, F. V. (1963). Human engineering and psychology. In S. Koch (Ed.), Psychology: A study of science (Vol. 5, pp. 831-907). New York: McGraw-Hill.

ACKNOWLEDGMENT

This research was funded by Air Force Contract F33615-86-C-0553 and does not necessarily reflect Air Force policy or opinion.

SIMULATION OF THE HUMAN-TELEROBOT INTERFACE

Mark A. Stuart and Randy L. Smith
Lockheed Engineering and Sciences Company
2400 NASA Road 1
P. O. Box 58561
Houston, Texas 77258

ABSTRACT

A part of NASA's Space Station will be a Flight Telerobotic Servicer (FTS) used to help assemble, service, and maintain the Space Station. Since the human operator will be required to control the FTS the design of the human-telerobot interface must be optimized from a Human Factors perspective.

Simulation has been used as an aid in the development of complex systems. Simulation has been especially useful when it has been applied to the development of complex systems. Simulation should ensure that the hardware and software components of the human-telerobot interface have been designed and selected so that the operator's capabilities and limitations have been accommodated for since this is a complex system where few direct comparisons to existent systems can be made. Three broad areas of the human-telerobot interface where simulation can be of assistance will be described. The use of simulation can not only result in a well-designed human-telerobot interface, but it can also be used to ensure that components have been selected to best meet system's goals and for operator training.

INTRODUCTION

The Space Station is a NASA project which, when completed in the mid-1990's, will function as a permanently manned orbiting space laboratory. A part of the Space Station will be a remotely controlled Flight Telerobotic Servicer (FTS). The FTS, a project led by NASA's Goddard Space Flight Center, will be used to help

assemble, service, and maintain the Space Station and various satellites. The use of the FTS will help ensure the safety and productivity of space-based tasks normally accomplished by astronauts performing outside the pressurized spacecraft. For the short-term, control of the FTS will be dependent primarily on the human operator. Since the human operator will be a part of the telerobotic system, then it is important that the human-telerobot interface be well-designed from a Human Factors perspective. It is critical that the components of this interface be designed so that the human operator's capabilities and limitations are best accommodated for within the structure of specific task requirements. To emphasize the importance of a well-designed human-telerobot interface, one study found that simply the selection of an appropriate control device, based upon the operator's capabilities and the requirements of the task, can more than double the productivity of the telerobotic system (O'Hara, 1986).

With the system development process becoming more complex and expensive, more emphasis is being placed on the evaluation of systems during early stages of the development cycle. The design of systems that include human operators is especially complex because determining overall systems performance is dependent upon the interaction of the human operator, hardware components, and software components (Chubb, Laughery, and Pritsher, 1987). Adequately evaluating the performance of a system during the design cycle is becoming increasing more difficult when using the static evaluation tools traditionally available to the

Human Factors Engineer, such as job and task analysis (Geer, 1981). It is becoming more common for systems developers to use simulation as a design tool instead of hardware models (Gawron and Polito, 1985) and for Human Factors Engineers to use simulation to enhance the use of static evaluation tools. This is because more sophisticated analysis tools are needed that will allow a controlled evaluation of the human operator/hardware components/software components interaction (Chubb, Laughery, and Pritsher, 1987).

This paper will cover the various uses of simulation, the elements of the human-telerobot interface, and how simulating the human-telerobot interface on the Space Station will result in a better designed system. Before focusing the discussion specifically to the simulation of the human-telerobot interface, it will be useful to briefly define simulation and to cover the major uses of system simulation -- independent of the type of system that is being simulated. There will then be a discussion of the areas of the human-telerobot interface and how simulation can contribute to a better designed user interface from a Human Factors perspective.

USES OF SIMULATION

Simulation is the process of imitating or duplicating the actions or processes of some system in a controlled environment (Arya, 1985). Emphasis should be placed on the word "controlled." System simulation, either hardware, computer, or a combination of the two, has been used for decades. This paper will describe four major uses of simulation. One use of simulation is to study the effectiveness of various hardware/software components on overall system's performance. The advantages of using simulation within this context are *cost* -- it is cheaper to simulate a system than it is to build one; *time* -- simulating a system is usually faster than building it; *feasibility* -- because of the size and complexity of some systems, it is not possible to evaluate them in the real world, therefore, simulation serves the function of systems verification; *safety* -- some systems operate in dangerous environments and can only be evaluated safely with the use of

simulation; and *prediction* -- with the use of simulation, a system's performance and processes can be speeded up so that future behavior can be predicted (Arya, 1985).

A second use of simulation is to determine the effects of various hardware/software components on simulated human performance. This approach utilizes mathematical models of human performance to assist the simulation process. In this, as well as the approach mentioned above, man-in-the-loop is not a part of the evaluation.

A third use of simulation is to investigate the effects of various hardware/software components on actual human performance. This approach can be taken in an attempt to match systems components and operator capabilities and limitations in order to ensure optimal systems and operator performance. This approach can be taken in an attempt to add greater fidelity, and thus, external validity to the data that are gathered in the analysis.

The last use of simulation to be addressed in this paper is to train operators to eventually use a real-world system. The major benefits of simulation as a training aid are in the areas of *scheduling* -- training is not affected by weather or the need to perform operational missions; *cost* -- simulator training is significantly less expensive than prime system training; *safety* -- reduces the exposure of operators and the prime system to the hazards of the operating environment; *control of training conditions* -- control of environmental and human interaction conditions that may be a part of the operating environment; *learning enhancement* -- system malfunctions and environmental conditions can be included in the training; and *performance enhancement* -- inclusion of critical missions that are difficult to train for in the real world (Flexman and Stark, 1987).

As the above list indicates, simulation has significant usage as an aid in the development of systems. It can have even greater significance in the design and development of novel systems -- systems that have never existed before and where few direct comparisons to existent

systems can be made. The human-telerobot system that will be used on the Space Station is such a novel system.

Even though industrial robots and teleoperators are heavily used in such areas as the nuclear industry and in underwater activities, there are major differences between these applications and the telerobot system to be used on the Space Station -- one of these being the zero-gravity factor. There is also a limited number of direct comparisons which can be made from the Remote Manipulator System (RMS) used on the Space Shuttle and the proposed telerobot system. The review of the literature concerning these systems has provided answers to some important design issues, but there are major limitations to how far these data can be generalized to the human-telerobot interface on the Space Station.

It is thus proposed that the use of simulation in the design and development of the human-telerobot interface on the Space Station will be very beneficial. Simulation should serve as an aid in the selection and design of hardware and software components to ensure maximum, error-free performance. Simulation should be worthwhile especially for its ability to simulate the effects of zero-gravity on performance. This can be accomplished in underwater conditions (e. g., Weightless Environment Test Facility at Johnson Space Center) and when people are exposed to momentary weightless conditions which can occur in certain aircraft (e. g., KC-135). Operator performance at manipulation tasks while in a one-gravity environment may well not be generalizable to weightless states. Simulation of the interface should also have the benefit of helping engineers to detect flaws in the design of components of the interface which would adversely affect system and/or operator performance. It is obviously important that any mistakes of this type be detected early and far before the design is finalized or manufacture of the system has occurred.

INFORMATION NEEDS OF THE OPERATOR

There are three broad areas of the human-telerobot interface where simulation can

be of assistance: operator information needs, control devices, and workstation layout. These three areas and examples of various components are listed in Table 1. The information needs of the operator will vary depending upon the tasks to be performed. The operator will need information concerning the location and orientation of the telerobot in space, the health status of the telerobot, visual feedback from the viewing system, the status of any transportation devices, the status of the workpiece, and the status of the hardware in the control workstation.

TABLE 1

Three areas of the human-telerobot interface and example components

1. Information needs of the operator
• Location of telerobot
• Status of transportation devices
• Status of workpiece
• Status of workstation
• Force feedback
• Visual feedback
Camera position and number
Spatial orientation of image
Monitor type, placement, and number
Illumination
2. Control devices considered
• Miniature master controllers
• 3 or 6 degree-of-freedom hand controllers
• Exoskeleton controllers
• Head-slaved controllers
• Dedicated switches
• Programmable display pushbuttons
• Voice commanded systems
• Computers
3. Telerobot workstation
• Hardware layout
• Software layout
• Restraint systems

Regarding visual feedback, the visual system may well be the single most important source of information for the operator (Smith and Stuart, in press). Some of the issues related to the visual system are concerned with camera

position and number, the spatial-orientation of the image presented to the operator, and monitor type, placement, and number. For example, when performing a remote manipulation task in real time, the operator can view the remote scene either by looking through a window, or with the use of cameras. For most of the tasks that will be performed in space, a direct view of the working area will either not be available, or will not provide the necessary visual cues for teleoperation. Therefore, cameras will provide the primary mode of feedback to the operator concerning manipulator position, orientation, and rate of movement. Operators normally use the body of the manipulator as a reference point when making control inputs, but, if the Space Station's external cameras are placed such that the camera view is not normal to the manipulator (normal refers to placement behind the shoulder of the arm) then the visual feedback will be spatially displaced. Spatial displacement is an unfortunate consequence of attempts to provide complete visual information to the operator; however, when the camera placement is not normal it should be avoided if at all possible or compensation techniques should be employed such as referencing the control device to the perceived motion on a monitor.

Spatially displaced feedback can take on different forms: *angular displacement* -- view is displaced horizontally within the transverse plane or vertically within the median plane; *reversal* -- view is facing the arm; *inversion-reversal* -- view is upside down and is facing the arm; and *inversion* -- view is upside down with respect to the manipulator arm. The image can also be *temporally displaced* -- there are time delays in the visual feedback, as well as *size distorted* -- the image is enlarged or reduced from its actual size.

Spatial displacements adversely affect operator performance to varying degrees. Generally, they take on progressively more disturbance with angular displacement being the least disruptive and inversion displacement being the most disruptive. Temporal displacement interrupts the intrinsic temporal patterning of motion and causes severe disruptions in behavior. Much effort should be

extended to prevent its occurrence. Size distortions generally do not affect performance to a great extent (Smith and Smith, 1962).

Other visual system issues include how an operator will use multiple views of the task area and how operators can best use non-stereoscopic cues to depth perception. (The issue of stereo versus monocular viewing is still being discussed in the literature and will not be addressed in this paper.) Simulation of various task scenarios with human operators working within various hardware and software mockups, including sophisticated scene generation techniques, can serve as an aid in determining what types of information are needed and what types of information presentation enhancements should be used at various points within the sequence of task performance. An example of an information enhancement technique that simulation can investigate is the use of real-time moving graphics displays designed to help operators maintain their orientation while performing under potentially visually disorienting conditions. Other screen-viewing techniques should be investigated with the use of simulation in an attempt to avoid operator disorientation while performing manipulation tasks.

CONTROL DEVICES

Control devices will be used to control such things as telerobot activation, position, manipulators, end effectors, rate of movement, and the viewing system. Control devices being considered include manipulator controllers such as miniature master controllers with direct position control, 3 or 6 degree-of-freedom hand controllers using rate or force inputs, exoskeleton controllers using various position sensors to detect human arm configurations, head-slaved control, dedicated switches, programmable display pushbuttons, voice commanded systems, and computer displays with cursor-control devices which allow menu selections. Control device selection is important because it affects operator performance, workload, and preference. Computer simulated scenarios could be linked to the use of actual controllers to determine their

effects on operator performance across different manipulation tasks. The study of different controllers while performing various tasks could also be used as a means of determining whether the use of specific controllers is more muscle-fatigue inducing.

WORKSTATION DESIGN

The telerobot workstation consists of hardware elements, their interfaces, and the software that will allow the hardware to be used. The workstation is the point where the information and control inputs are made available to the operator. Just as with the selection of control devices, the workstation should be logically and functionally laid out to optimize operator performance and preference while minimizing workload and error rates. Simulation can help to determine optimal workstation layouts. A simple means of simulating the workstation layout is through the use of computer prototyping, but it is recommended that large-scale simulation be used as a means of designing and evaluating the telerobot workstation.

CONCLUSIONS

Many issues remain unresolved concerning the components of the human-telerobot interface mentioned above. It is then critical that these components be optimally designed and arranged to ensure, not only that the overall system's goals are met, but that the intended end-user has been optimally accommodated for. With sufficient testing and evaluation throughout the development cycle, then the selection of the components to use in the final telerobotic system can promote efficient, error-free performance. It is recommended that whole-system simulation with full-scale mockups be used to help design the human-telerobot interface. It is contended that the use of simulation can facilitate this design and evaluation process. The use of simulation can also ensure that the hardware/software components have been selected to best accommodate the astronaut, instead of the astronaut having to make performance accommodations for the hardware/software components that have been selected.

As was mentioned above, there are other advantages to simulating the human-teleoperator interface than simply serving as an aid in the selection and design of hardware/software components so that operator performance is optimized. Systems developers can also use the simulation system to test whether or not hardware components meet overall systems goals, and the simulation system can be used for subsequent training of the astronauts who will use the actual system.

ACKNOWLEDGEMENTS

Support for this investigation was provided by the National Aeronautics and Space Administration through Contract NAS9-17900 to Lockheed Engineering and Sciences Company.

REFERENCES

- Arya, S. (1985). Defining various classes of simulators and simulation. In J. S. Gardenier (Ed.), *Simulators* (pp. 36-38). La Jolla, California: Simulation Councils, Inc.
- Chubb, G. P. Laughery, Jr., K. R., and Pritsher, A. A. B. (1987). Simulating manned systems. In G. Salvendy (Ed.), *Handbook of human factors* (pp. 1298-1327). New York, New York: John Wiley and Sons.
- Flexman, R. E., and Stark, E. A. (1987). Training simulators. In G. Salvendy (Ed.), *Handbook of human factors* (pp. 1012-1038). New York, New York: John Wiley and Sons.
- Gawron, V. J., and Polito, J. (1985). Human performance simulation: combining the data. In J. S. Gardenier (Ed.), *Simulators* (pp. 61-65). La Jolla, California: Simulation Councils, Inc.
- Geer, C. W. (1981). *Human engineering procedures guide* (AFAMRL-TR-81-35). Wright-Patterson Air Force Base, OH: Air Force Aerospace Medical Research Laboratory.

O'Hara, J. M. (1986). *Telerobotic work system: Space-station truss-structure assembly using a two-arm dextrous manipulator* (Grumman Space Systems Report No. SA-TWS-86-R007). Bethpage, NY: Grumman Space Systems.

Smith, K. U., and Smith, W. M. (1962). *Perception and motion: An analysis of space structured behavior*. Philadelphia, Pennsylvania: W. B. Saunders Company.

Smith, R. L. and Stuart, M. A. (in press). Telerobotic vision systems: The human factor. In *Proceedings of the Instrument Society of America (ISA) Conference*. Houston, Texas: Instrument Society of America.

HUMAN INTERACTIONS WITH INTELLIGENT, AUTONOMOUS SYSTEMS

Roger Shapell
Martin-Marietta INCS

(Paper not provided by publication date.)

ORIGINAL PAGE IS
OF POOR QUALITY

MAN-SYSTEMS REQUIREMENTS FOR THE
CONTROL OF TELEOPERATORS IN SPACE

Nicholas Shields, Jr.
Human Factors Consultant
CAMUS, Incorporated
2704 Churchill Drive
Huntsville, Alabama 35801

Abstract

The microgravity of the space environment has profound effects on humans and, consequently, on the design requirements for subsystems and components with which humans interact. There are changes in the anthropometry, vision, the perception of orientation, posture, and the ways in which we exert energy. The design requirements for proper human engineering must reflect each of the changes that results, and this is especially true in the exercise of control over remote and teleoperated systems where the operator is removed from any direct sense of control.

The National Aeronautics and Space Administration has recently completed the first NASA-wide human factors standard for microgravity. The Man-Systems Integration Standard, NASA-STD-3000, contains considerable information on the appropriate design criteria for microgravity, and there is information which is useful in the design of teleoperated systems. There is not, however, a dedicated collection of data which pertains directly to the special cases of remote and robotic operations.

This paper deals with the design considerations for human-system interaction in the control of remote systems in space, briefly details the information to be found in the NASA-STD-3000, and argues for a dedicated section within the Standard which deals with robotic, teleoperated and remote systems and the design requirements for effective human control of these systems in the space environment, and from the space environment.

Introduction

The history of manned space flight is filled with the scientific and exploratory accomplishments of humans and demonstrations of our productivity in the orbital environment. During the Skylab era, were it not for the corrective measures taken by the first manned mission to Skylab, the program

would have been lost. The several satellites which have been recovered, repaired and returned to orbit by Shuttle crew members is testimony to the key position that humans hold in carrying out our successful space program. The Apollo Program sent men to the Moon and returned them, and the results of their exploration, as well as their impressions of our planet from a new vantage point.

In the next decades, we will return to the moon and venture out to Mars. We will orbit the Earth in a permanently occupied Space Station, and begin the colonization of our solar system. We will do all of this based on our experiences and successes of past missions and our desire to comprehend the Universe around us.

The lessons and legacies of our manned space flight experience, space systems research and human productivity in space have been compiled in the NASA-STD-3000, Man-Systems Integration Standard, the first NASA wide design guide for man-systems in space flight. This four volume set of design guidelines presents the design considerations and requirements for the effective employment of humans in space. The table of contents reflects the human engineering issues which must be addressed in order to support humans in space, both inside of spacecraft (intravehicular activity) and outside of spacecraft (extravehicular activity). In point of fact, the two precedent human engineering guidelines for space flight programs were divided along the EVA and IVA roles in space. The MSFC-STD-512A is a very detailed treatment of the IVA issues, while the JSC 10615 dealt with the EVA design considerations and requirements. The contents and philosophy of these two useful guidebooks have been combined and superceded by the NASA-STD-3000. But is the support of humans in space the only way to effectively conduct space exploration and operations?

ORIGINAL PAGE IS OF POOR QUALITY

Many of the research programs that are sponsored by NASA in the areas of robotics and teleoperation suggest that direct human presence and intervention are not the only means by which we can explore and manipulate the space environment around us. The Marshall Space Flight Center has conducted research in teleoperated systems since the late 60's (1). The Jet Propulsion Laboratory has developed and launched numerous unmanned explorer spacecraft and the Orbital Maneuvering Vehicle is being developed to augment the role of humans in space without exposing them to the hazards and risks of the space environment. The Goddard Space Flight Center is developing the Flight Telerobotic Servicer for remote operations and the Space Station will have a Mobile Servicing Center for the conduct of remote activities at the Space Station. But what of the human operators who will be responsible for the management and operation of these teleoperated and robotic systems? Where are the design criteria which we will employ in the effective integration of human capabilities and those of machines for robotic space operations?

Background

During the definition stages of the Man-Systems Integration Standard every effort was made to identify the categories of experience which we had gained over the past twenty-five years of human space flight. The organization of the standards follows very closely the organization of conventional human engineering and applied psychology texts, but the bibliography and research literature on which the standards are based is unconventional, coming principally from space flight data files and reports. Consequently, we find subject matter titles such as vision, anthropometry, human performance, grip strength, etc. filled with data which is not familiar to human factors specialists who deal only with Earthly design concerns. Alteration of posture, visual capability, spatial orientation and biochemical components of the human are a few of the significant differences experienced as the result of space flight and the effects of microgravity.

During the development of the Man-Systems Integration Standards, there was considerable discussion concerning the treatment of extravehicular activity design data and requirements. "EVA is a special set of operations requiring a special treatment in the development of design standards", was one of the arguments. Certainly, the fact that the human assumed the shape of the space suit, that without the space suit there could be no EVA, and that the boundary of the space suit was the envelope of design interest, were all radically different factors than those which have to be considered for IVA, or shirt sleeved

operations conducted in a pressurized space craft. On the other hand there was the argument for an integrated design standard which dealt with space flight issues as though there were not significant differences among the several classes of activities. "Put the EVA requirements and considerations in a sub-paragraph of the topic of interest", went the argument, assuming that space flight activities are space flight activities.

The recognition that EVA is a significantly different means of conducting space operations is evident in the dedicated chapter detailing EVA design requirements in the NASA-STD-3000. The organization of this chapter follows the organization of the standard itself, but the details pertain to the special design constraints associated with EVA. So, the argument for a separate chapter prevailed, but another argument was lost, that for a chapter dealing with the special design constraints associated with robotics and teleoperation as a means to carry out space operations.

Space Automation, Robotics and Teleoperation as a Special Class of Space Operations

The technology to perform remote operations with humans as the primary controller or supervisor is well demonstrated on a daily basis in chemical processing plants, electricity generating plants, undersea exploration and operations and in steel processing facilities to cite only a few examples. Human operators visually inspect and monitor, manipulate and order, control movement and orientation of remote systems as though they were actually in the remote environment. To do so requires special technology and specific information be made available to the operator. The content and format of this information and the control and display requirements to manage this technology are not always the same as they are in a conventional, direct management work situation. The support requirements for remote vehicles are also different from those which are managed directly by humans. Just as handrails, handholds and work restraints are required for the conduct of EVA, teleoperated systems require special design consideration to support the man-machine symbiosis. The issue is, where do we go to find these special design considerations and requirements for remote space operations? If both remote operations and space operations are special classes of activities performed by humans and machines the question then becomes, is there a human engineering design data base for remote space operations?

Remote Vision

Using direct visual apprehension, the human is able to detect targets as small as .01 arc minutes, able to perceive variation

ORIGINAL PAGE IS OF POOR QUALITY

among hundreds of colors, estimate distances using stereoscopic, as well as monoscopic, cues and detect motion in the visual field. The current state of video technology does not take full advantage of the human's visual capability and, consequently, some design compromises have been made which have been shown to have a negative effect on system performance. Black and white video offers higher resolution than current color television, but at the expense of losing the advantage of the information conveyed by colors in the remote scene (2). Stereoscopic television systems provide a means of perceiving depth in a visual display, but usually at the cost of reduced frame rate, reduced field of view, constraints on head movement or reduced luminance (3). Even in the best systems, sensor and display technology combine to limit the resolution of the remote scene to 3 or 4 arc minutes, or require a prohibitively large bandwidth for signal transmission, especially for space applications. And the field of view that is available from most display systems is greatly less than the normal field of view that we use to comprehend the environment on a day-to-day basis.

On the other hand, video technology permits us to combine graphical data with visual scenes, augment displays with computer generated information, build synthetic displays which can be used to rehearse an activity before executing it, focus on a specific point in the visual field, enlarge or reduce the field of view, greatly magnify an object, and actually insert a visual probe in spaces where we would otherwise be unable to see. In some systems, multiple cameras and displays can afford a forward, as well as a backward, view of the remote environment. And in others, we can enhance a visual scene through computerized reconstruction to provide a representation of the remote environment that would otherwise be meaningless. There are new technologies such as fiber optics, head up displays, helmet mounted displays and virtual image displays which can be employed in the control of remote systems, and we are coming to understand how and where this technology can be effectively used. What is not fully understood, from a functional standpoint, are the effects on operator performance that this technology has. What is lacking is a description of what we do know about human performance and remote vision and system performance as they are applied to robotic space applications.

Remote Manipulation

Through the use of his hands, the human is able to sense small forces or exert gripping forces for a short period in excess of 100 pounds. Using direct touch, the operator can manipulate objects that are out of view. He can make quick and delicate motions to

change items in the direct environment, or make quick and forceful motions and crush a concrete tile with his fist. The operator can sense differences in mass by comparing two objects held in his two hands. And by picking up a tool, he can multiply his capacity to manipulate and alter the environment within his reach. The ergonomics of manual dexterity, fatigue, operating errors, and the tactile senses are well studied and documented in conventional human engineering texts and design guidelines. The human requirements for control and management of remote manipulation are not so well understood or documented. For space operations there is not a formal body of knowledge to which a system designer can turn for design requirements and guidelines.

As a means of manipulating and changing the remote environment, space teleoperators are usually envisioned with a manipulator arm, at least one and more often with several. The terminal effector is generally drawn as a clamp or multi-fingered hand, or in the case of a teleoperated Mars soil sampler, a simple scoop. The Shuttle Remote Manipulator System (RMS) has a terminal effector which can accept only specially prepared grapple fixtures in order to handle remote payloads. More advanced manipulator systems for space, such as the Flight Telerobotic Servicer and the Orbital Maneuvering Vehicle are being designed with more manipulating capability and a more flexible terminal effector than are available on the RMS. However, the design requirements to take full advantage of the capabilities, and avoid the limitations, of the human operator in remote manipulation are not yet fully developed.

The minimum detail required to support the design of systems for remote space manipulation will require an understanding of the effects of employing a particular end effector, the type of articulated arm, the control algorithm and the control devices used by the human to accomplish the remote task. A significant change in any of these components has been demonstrated in the laboratory to have a change in the overall system performance (4).

Design considerations and requirements for remote space manipulation should include the use of general purpose effectors such as grasping fingers, opposed clamps, parallel jaws and other, near anthropomorphic approaches. The design considerations for specialized effectors such as terminal tool kits, inflatable end effectors, tactile probes, and capture and docking devices should be detailed in a design handbook. Each time we want to employ a remote manipulator end effector system, we should not have to design it from scratch and for only one mission or application, but rather we should be able to refer to a class of

ORIGINAL PAGE IS OF POOR QUALITY

demonstrated designs and the effects they have on human performance.

As we move up the manipulator from the effector, we will want to know the consequences of employing a particular style and design of manipulator arm. There are demonstrated performance differences among classes of arms with respect to various tasks. Telescoping arms, for example, are commanded to a specified point in less time than are multi-jointed, articulated arms (5). This might be useful information in the design of a capture and docking device for space teleoperators, where the teleoperated arm is required to reach out and get a secure hold on a specific capture fixture on a satellite. The system designer might want to know the performance differences between the control of anthropomorphic and non-anthropomorphic arm designs, if there are any. And what are the effects on the operator of adding more degrees of freedom to an articulated arm? Can the more complex design produce a higher level of performance, or does the operator become confused in the control of multiple degrees of freedom and consequently, overall system performance declines?

One of the recurring issues in teleoperated space systems is the number of arms that one operator can and should control. One way to avoid the design solution of providing the maximum imaginable number of arms required to perform the task and then forcing the operator to contend with the simultaneous control of these is to include the human performance design requirements in the system design, but where do we look for such requirements?

Between the physical manipulator with its end effector and the human operator of a space teleoperator there is the control algorithm. What approaches to control software produce the best results for space teleoperation? Do operators perform a class of tasks better when they have tip position control, or joint control, or does it make any difference? Is system performance changed when the software executes specified routines rather than having the operator have to perform them? At what rates should a control algorithm permit a manipulator to execute a task if, at any time, the intervention of the human operator is likely to be required to manage unforeseen circumstances? The designer of space teleoperators should be able to consult a design guide which addresses these issues, if not answer them.

The control system by which the human operator manages the remote manipulator in space might be a manual controller, or a voice controller. The manual controller might be one or two handed, a joystick or trackball, exoskeletal, replica, force reflecting or position commanding. But

which is better and which is best?

A serious attempt has been made by the National Bureau of Standards to quantify the performance criteria for measuring manipulator capabilities, and to standardize the devices and methods used to evaluate manipulator systems, so the data bases are available or under construction (6). It is really a matter of getting the information into the hands of the design engineers in a format that is useful, and with full recognition of the human operator as a central feature of the teleoperated manipulating system.

Workstation Design for Remote Operations

When we consider the design of workstations for remote space operations we are confronted with two populations of operators, those who operate from the microgravity environment and those who operate from ground based control stations.

They are exclusive populations in terms of anthropometry and operational requirements. The designer of teleoperator workstations should have the advantage of what has been learned about the design constraints which apply to both of these populations. There has been significant research and design work done for both the terrestrial and the microgravity workstations, consequently the issue of work station design is a less pressing one if the designer is familiar with the requirements which suit both populations.

Again, the required study has been accomplished. We know how to design Earth based workstations which complement the operator's ability to perform remote tasks, and we know how to take advantage of the microgravity environment, its effects on human performance, and design workstations to accommodate to these factors. What is needed is the incorporation of these data in a dedicated chapter of the Man-Systems Integration Standard which deals with teleoperation and automation. Here the designer could review the postural and anthropometric changes that take place as a result of living and working in space, the increase in stature and the effective decrease in operational posture. The designer could review the effects on vision and visual perception which accompany an environment which does not filter and refract light through a thick atmosphere. He could review the requirements for operator restraint at a workstation and determine if the restraints would accommodate a spring loaded, force reflecting hand controller without having it push the operator away from the workstation as control forces, and equal and opposite reactive forces, are transferred to and from the hand controller.

System Induced Factors

Working from Earth to control a space based servicing teleoperator may involve distances of only 400 or 500 kilometers. However, the transmission and relay of commands and feedback on such low Earth orbital exercises may be as much as 2 or 2.5 seconds as a function of network delays. With the use of significant ground networks it could be even more. What are the effects on the human operator of having such a delay in the control loop? Are the effects different for different periods of delay? Are there differences in system performance when the delay is random and unpredictable, a function only of the network cycle times? Are there design solutions which have been shown to be effective in compensating for control loop delay? Control loop time delay has been the subject of several recent NASA programs, and will continue to be a topic of interest as more robotic space vehicles are placed in service (7). The issue remains, however, as to the best means to provide the research findings and design considerations to the system designer. We should not expect, as information consuming and processing animals, that every designer should be aware of the study results concerning all of the subsystems with which the human is required to interact with during a teleoperated mission. These data should be made available in a centralized data base detailing the response of humans to remote systems technology.

Summary and Invitation

Each year the NASA-STD-3000 is reviewed by a government and industry advisory group, and critical information is added, modified and edited to make the standard more reflective of the changing technology, new research findings and program requirements. The NASA Johnson Space Center is responsible for maintaining the critical comments and reviewing them for incorporation in the standard. These comments are classified into four categories as follows:

1. Introductory, explanatory and clarifying statements which introduce the topic to the reader. For a space teleoperations section this would include a definition and discrimination of teleoperators, robots, artificially intelligent machines, automata and the like. It would provide a description of the general system to include the orbiting or roving machine, the relay and transmission system, the effectors on the machine and the human operator as an element in the control loop.

2. Design considerations and comments. These are the salient points to consider in the design of a human operated system, although the considerations may not give strict rules to follow. For

teleoperated systems, the design considerations might include the degree of telepresence appropriate to the control system, the variety of hand controllers which are available for control of teleoperators and a general discussion of the differences among them. In this section the issues of time delay, color coding display formats and arrangements would be dealt with as items which the designer must take into consideration as he or she begins to define the remote system to meet his or her special systems' requirements.

Very often the information contained in the considerations section is more important than information contained in other of the sections, but because it serves as a menu of options, it is usually not detailed and specific enough to tell the designer what to do, just what to consider.

3. Design requirements is the third section for each topic in the standard. It is in this section where engineers and designers find the detailed requirements which must be met in order for the system in question to meet the demands made on human operated space vehicles. Where possible, the requirements are specified in quantitative terms, usually within a design range. Variations from any of these requirements calls for a review and approval of the design variation. For space based teleoperators the requirements might call for a fixed period of time which an operator can work without relief, or state that control inputs shall not be capable of accidentally damaging the craft, or that display resolution shall be greater than 1 arc minute. They would probably state the minimum display rate, signal-to-noise ratio and contrast and display brightness. Concerning the use of flight controllers and manipulator controllers, the requirements would specify force and torque limits and the number of degrees of freedom which can be controlled by an operator. The requirements for space teleoperators will probably seem overly restrictive, but they will ensure against system failure and damage to adjacent structures. The requirements are those items which must be satisfied in order to ensure an appropriate allocation of authority and autonomy between the human and the machine.

4. Design examples and solutions is the fourth section of each of the topics covered in NAS-STD-3000. Here, proven space designs are presented, not as the answer to a designer's dream, but as historically successful solutions to problems encountered in space systems. For space based teleoperators and robots this section would include the Mars lander, the Soviet lunar rovers, the Shuttle Remote Manipulator System and other extant examples. It might also supply design solutions from very near term programs such as the Flight Telerobotic

Servicer or the Orbital Maneuvering Vehicle if they advance the state of the art or understanding beyond that provided by historical missions.

As the role of teleoperators and robots becomes more wide spread in the space environment, and as NASA and the Department of Defense come to rely on them more, there will be a clear requirement to develop a dedicated human engineering design standard for telerobot systems. Those of us who are interested in seeing the effective application of this technology can contribute our concerns and knowledge to such agency wide standards as NASA-STD-3000. First, we can request to be included in the next Government and Industry Advisory Group meeting, and second, we can send recommendations concerning the incorporation of man-systems/remote systems data into the existing standard. The Johnson Space Center is responsible for maintaining the Standard, and comments and considerations can be forwarded to Mr. Cletis Booher, SP3/Man-Systems Integration Standards, NASA-Lyndon B. Johnson Space Center, Houston, TX 77058. It is hoped that in the next few years, through the efforts of participants in symposia such as SOAR and the Robotics Industry Association that we will be able to define and contribute a body of knowledge which will encourage the application of automata, robots and teleoperators to the operations of our space program.

References

1. Shields, N.L., Jr., et.al. Human operator performance of remotely controlled tasks: a summary of teleoperator research conducted at NASA's George C. Marshall Space Flight Center between 1971 and 1981. NASA Contractor Report, Marshall Space Flight Center, Alabama 35812, 1982.
2. Shields, N.L., Jr. and Fagg, M. F. Analysis and selection of a remote docking simulation visual display system. NASA Contractor Report, Marshall Space Flight Center, Alabama 35812, 1984.
3. Shields, N.L., Jr. Stereo video and display systems. Conference Proceedings of the Rendezvous and Proximity Operations Workshop. Lyndon B. Johnson Space Center, Houston, Texas, 1985.
4. Kirkpatrick, M., et.al. Manipulator system performance measurement. Proceedings of the Second National Conference on Remotely Manned Systems. University of Southern California, 1975.
5. Malone, T.B., et.al. Manipulator system man-machine interface evaluation program. NASA Contractor Report, NAS8-28298, 1974.
6. National Bureau of Standards. Proceedings of the NBS Workshop on Performance Evaluation of Programmable Robots and Manipulators. Annapolis, Maryland, 1975.
7. Paulukaitis, K.R., et.al. Remote robotic servicing of space systems with time delay. Essex Corporation, 1986.

**ORIGINAL PAGE IS
OF POOR QUALITY**

ORIGINAL PAGE IS
OF POOR QUALITY

INTEGRATION OF A COMPUTERIZED TWO-FINGER GRIPPER FOR ROBOT WORKSTATION SAFETY

John E. Sneckenberger, Professor
Kazuki Yoshikata, Graduate Student

Mechanical and Aerospace Engineering
West Virginia University

ABSTRACT

A microprocessor-based controller has been developed that continuously monitors and adjusts the gripping force applied by a special two-finger gripper. This computerized force sensing gripper system enables the endeffector gripping action to be independently detected and corrected. The gripping force applied to a manipulated object is real-time monitored for problem situations, situations which can occur during both planned and errant robot arm manipulation. When unspecified force conditions occur at the gripper, the gripping force controller initiates specific reactions to cause dynamic corrections to the continuously variable gripping action.

The force controller for this intelligent gripper has been interfaced to the controller of an industrial robot. The gripper and robot controllers communicate to accomplish the successful completion of normal gripper operations as well as unexpected hazardous situations. An example of an unexpected gripping condition would be the sudden deformation of the object being manipulated by the robot. The capabilities of the interfaced gripper-robot system to apply workstation safety measures (e.g., stop the robot) when these unexpected gripping effects occur have been assessed.

INTRODUCTION

The widespread application of robots has created a need for endeffector devices which can sense the force applied to handled objects. Other handled object characteristics such as its shape are also often required because the robot has to deal with several kinds of object materials of different texture.

Although these kinds of gripper capabilities are absolute necessities for so-called human equivalent robots, the mentioned endeffector functions can have specific purposes for some industrial applications. In this project, the proposed industrial gripper force sensor system can be fully utilized for handling part objects with only limited feedback information from the endeffector because:

1) The user enters the approximate gripper force that seems suitable for gripping a particular object while the robot performs a specified task.

2) If the robot's smart gripper controller then notices that the object cannot be safely handled within the program specified force bounds due to prescribed task acceleration and deceleration characteristics, it will stop the robot and wait for another user command that it recognizes to be more suitable for the particular part handling assignments, force-wise and robot movement-wise.

Thus, the robot can avoid improper and unsafe part handlings.

GRIPPER DESIGN

In manipulating and moving objects with robots, force sensors in the gripper can provide important feedback information.

A smart gripper force sensor system using strain gages was designed to permit the handling of very delicate objects using a two-finger gripper (see Figure 1).

The details of the gripper fingers and force sensor design will be described in a paper to be presented at the International Conference on Ergonomics of Advanced Manufacturing and Hybrid Automated Systems this August 15th.

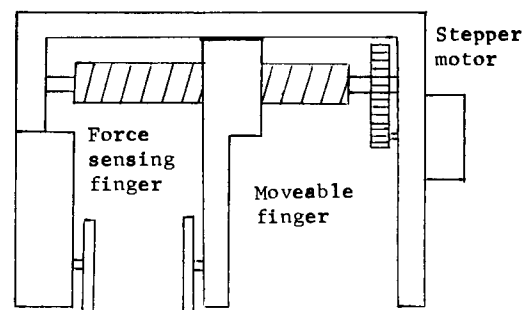


Figure 1. Two-Finger Gripper Hardware

SMART GRIPPER CONTROLLER

The signal from the gripper's sensor is amplified by a strain gage amplifier which sends a DC output voltage to the input of an A/D converter (see Figure 2). The voltage is processed through an A/D converter and sent to an 8-bit NEC Z80A computer through a 8255A PIO interface, where interface programs can be written in either Basic or Assembler languages. Also the stepper motor for the gripper drive is driven by a stepper motor driver board through another PIO interface. Further, the NEC computer communicates with GE P-50 industrial robot through another PIO interface.

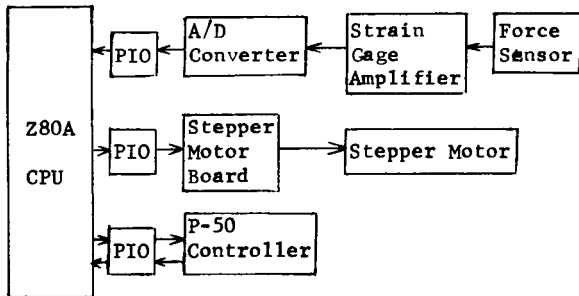


Figure 2. Control Computer Interface

GRIPPER CONTROL LOGIC

During robot operation, the two-finger gripper maintains the user specified force; that is, the forward and reverse rotations of the stepper motor are controlled by the output signals from the smart gripper controller. As the gripper force approaches the specified force, the forward and reverse rotations of the stepper motor are slowed down to one-third of full speed.

In this developed smart gripper system, improper part handling force was defined as $F < 1/2 W$ or $F > 3/2 W$, where W is the user specified force. If the measured force exceeded the defined safe range during robot task operation, the robot automatically stops and waits for the user to enter another specified force (see Figure 3). Thus, the smart gripper controller provides a robot safety scheme in its handling of objects.

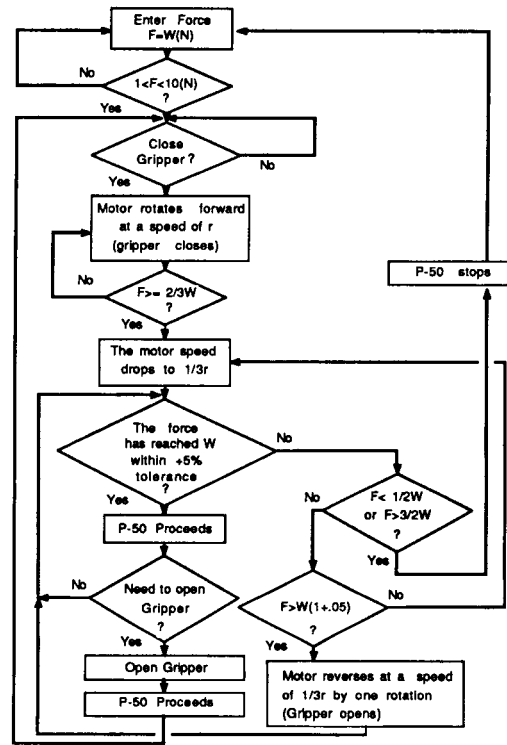


Figure 3. Flow Chart of Force Control Logic.

SMART GRIPPER OPERATIONS AND CIRCUITRY

The operational logic and interface communications for the smart gripper controller will be described by means of the following pick-and-place illustration.

Grasp (Position B)

When the robot gripper reaches its part grasp location (position B) from its home position, after 1 sec. the P-50 controller output port 04 turns on (see step 01 in Table 1 and Figures 4 and 5). Thus, the I/O port A0 of the Z80A is then grounded and the Z80A starts its A/D conversion. The stepper motor then starts rotating, causing the gripper to close.

When the specified force is applied to the object, relay 1 closes (see step 02 in Table 1). The P-50 robot then proceeds to the part release location (position A) for the gripper.

Transfer (Move B → A)

While the object is being transferred, the stepper motor can rotate forward and reverse according to the continuous force feedback. If the force is too big or too small owing to some physical, etc., change, where a proper gripping operation is now impossible, then the P-50 robot stops and waits for another force input from the Z80A. Relay 2 is activated to initiate this servo-stop of the P-50 robot.

Release Location (Position A)

When the P-50 endeffector reaches its release location (position A), after 1 sec. the P-50 controller output port 05 is turned on (see Step 03 in Table 1). Then the motor reverses and the object is released by the gripper.

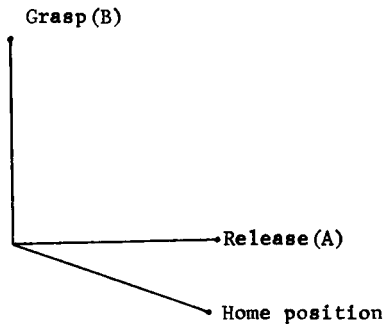
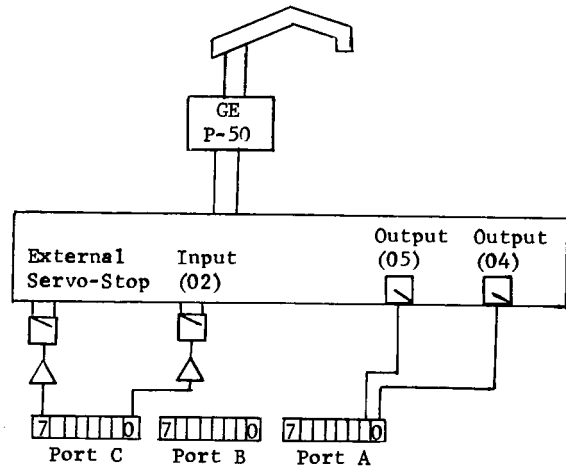


Figure 4. P-50 Operational Sequence with Safety Scheme.

Table 1. Control Logic Step Sequence

Step	P-50 Timer	P-50 Output	P-50 Input	Comments
00				Home Position
01	T1 ON (1 sec)	04 ON		P-50 moves to position B. After 1 sec, gripper closes to the specified force.
02			02 ON External servo stop	When the force reaches the specified value, the P-50 proceeds, maintaining the force. If the force exceeds the range: $F < (1/2) * W$ or $F > (3/2) * W$, P-50 stops. W is a specified force.
03	T2 ON (1 sec)	05 ON		When the P-50 reaches position A, after 1 sec the gripper opens.
04				End of Program



(Output port) (Unused) (Input port)

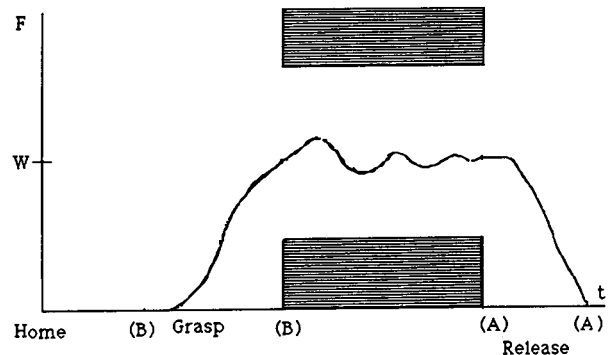
8255A I/O Ports of NEC Z80A Computer

Figure 5. Z80A Interface Connection

EXPERIMENTAL STUDY

Experiments were performed to learn how accurately the endeffector was able to handle deformable objects (see Figure 6). These pick and place tasks were conducted at various robot speeds.

Although complete numerical results have not yet been obtained, the designed and developed smart gripper has shown excellent observed performance in the handling of lightweight deformable objects (see Figure 7).



* Shaded areas define unsafe forces.

Figure 6. Example of Controlled Gripper Force.

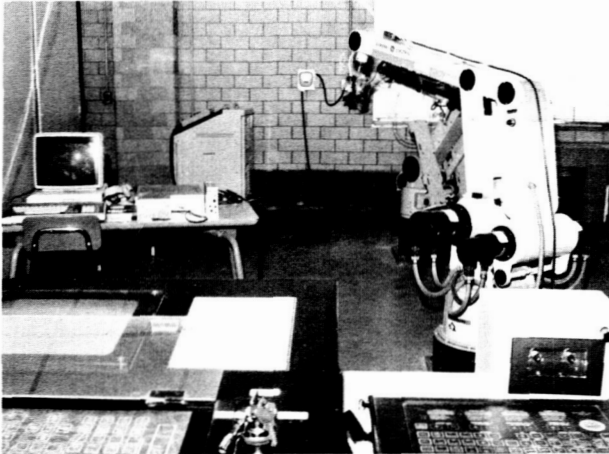


Figure 7. Experimental System.

CONCLUSIONS

This project has been conceptually successful in the development of a low cost gripper force sensor system that was able to handle lightweight deformable objects during a pick-and-place task with safety. This system was conceived and constructed with controller interface provisions that stops the robot if the gripper force exceeds a specified unsafe force range.

REFERENCE

1. "Tactile Sensors and The Gripping Challenge." IEEE Spectrum, August 1985, pp. 46-52.
2. "IBM's Robotic Gripper with Tactile Feedback." Epsilonics, Vol. 3, Issue 3, December 1983.
3. Antal k. Bejczy, "Robotic Hands on Ladder of Evolution." NASA Tech Briefs October 1987, pp. 16-18.
4. Yoshikata, K., "A Microcomputer Controlled Tactile Gripper Sensor for Robot Manipulator Endeffector Design." ASME International Computer in Engineering Conference, August 7- 11, 1983, pp. 97-101.
5. ELEHOBBY INTERFACE Manual 1, SENSOR & A/D, D/A CONVERTER MANUAL 4, STEPPING MOTOR Manual 3, ELEHOBBY Corporation, Fukuoka, Japan, 1986.
6. Kyowa Strain Gage Instrumentation Catalogue, Kyowa Corporation, Tokyo, Japan, 1985.
7. Shigley, Joseph Edward, Mechanical Engineering Design, 3rd edition, McGraw-Hill, pp. 27-56, 320-395, p. 678.

LOCAL POSITION CONTROL:

A NEW CONCEPT FOR CONTROL OF MANIPULATORS

Frederick A. Kelly

Mechanical Engineering - Engineering Mechanics Department
Michigan Technological University
Houghton, MI 49931

ABSTRACT

Resolved motion rate control is currently one of the most frequently used methods of manipulator control. It is currently used in the Space Shuttle remote manipulator system (RMS) and in prosthetic devices. Position control is predominately used in locating the end-effector of an industrial manipulator along a path with prescribed timing.

In industrial applications, resolved motion rate control is inappropriate since position error accumulates. This is due to velocity being the control variable. In some applications this property is an advantage rather than a disadvantage. It may be more important for motion to end as soon as the input command is removed rather than reduce the position error to zero.

Local position control is a new concept for manipulator control which retains the important properties of resolved motion rate control, but reduces the drift. Local position control can be considered to be a generalization of resolved position and resolved rate control. It places both control schemes on a common mathematical basis.

INTRODUCTION

Space presents a uniquely promising work environment for operations associated with research, manufacturing, and services in a number of fields of commercial importance [1]. Semiconductor, superconductor, and biological technologies and satellite servicing being some of the more promising areas of commercial uses of space.

Space also presents a uniquely unfamiliar and hazardous work environment for people. The micro-gravity and ultra-vacuum of low earth orbit, which is so promising in its technological uses, exposes astronauts to potentially dangerous situations. The use of advanced automation and robotics in space is seen as a way of reducing both the risks and the costs of space operations [2].

Astronauts excel at integrating sensory information, interpreting sensory information, and then using his judgment to make decisions as to how best to complete a task, even in the event of some unforeseen circumstance. An astronaut, however, can become overwhelmed by sensory information and not be able to perform effectively. Teleoperated manipulators, like the Shuttle remote manipulator system (RMS), are an extension of a person's sensing and manipulating capability to a location remote from him.

Automation and robotics will not lessen the importance of the astronaut, since fully autonomous operation in space is not possible in the foreseeable future. Rather, astronauts and automation need to be utilized each to their best advantage. Automation excels in quickly storing and recalling large amounts of data, computing, responding to signals, and in continuously monitoring many different tasks without being distracted. Sharing controlling between astronauts and automation can result in a system with greater capability than either operating alone.

The problem of sharing control of a manipulator between man and machine is of crucial importance for space and other remote applications of automation. Some directions in which automation and robotics research should evolve from this technology into more advanced telerobots are described by Sheridan [3]. Sheridan distinguishes between efferent (motor) and afferent (sensory) computer extensions to the human operator. This paper describes a new concept for control of manipulators, local position control. It is an efferent extension in that it unifies position and resolved rate control techniques and is also an afferent extension in that in some implementations it would reduce the sensory burden.

CONTROL OF MANIPULATORS

A common characteristic of manipulator control systems is that they must generate a trajectory for one or more appendages. A trajectory consists of two parts, a path and a displacement along that path. A teleoperator control system [4] is shown in Figure 1.

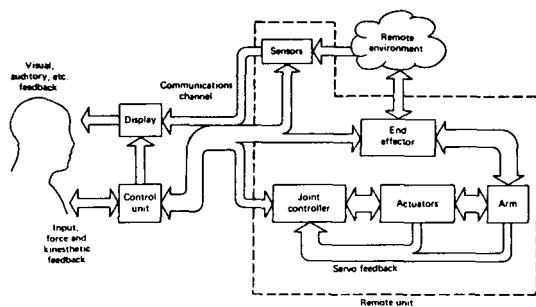


Figure 1 - Teleoperator Control System

Resolved motion rate control allows the operator to specify the velocity of the end-effector in directions resolved into a Cartesian coordinate system. It is one of the most frequently used methods of manipulator control and is used in the RMS and prosthetic devices. Here, the path generated and the path commanded diverge since rate is control variable. At any instant the input is commanding a velocity on the path generated without regard to the path commanded.

Resolved position control allows the operator to specify the position of the end-effector in directions resolved into a Cartesian coordinate system. It is commonly used in locating the end-effector of an industrial manipulator along a path with prescribed timing.

In industrial applications, resolved motion rate control is inappropriate since position error accumulates. In some applications, however, this property is an advantage rather than a disadvantage. It may be more important for motion to end as soon as the input command is removed rather than reduce the position error to zero.

LOCAL POSITION CONTROL

Local position control can be thought of as consisting of position control in a plane normal to the path and rate control along the path. The concept is shown graphically in Figure 2. For a path P between two points A and B, the actual position is point C at a particular instant in time t. Point C is in error since it is not on the path P. Under the local position control paradigm, the commanded position at time $t + \Delta t$ can be found by projecting C in a plane normal to the path to point D on the path and then advancing along the path to point E according to the desired time rate of change of displacement along the path. The next position can be expressed as

$$\underline{r}(t + \Delta t) = \underline{r}_{E/C} + \underline{r}(t) \quad (1)$$

where

$$\underline{r}(t) = \underline{r}_{C/O} \quad \text{and} \quad \underline{r}(t + \Delta t) = \underline{r}_{E/O}$$

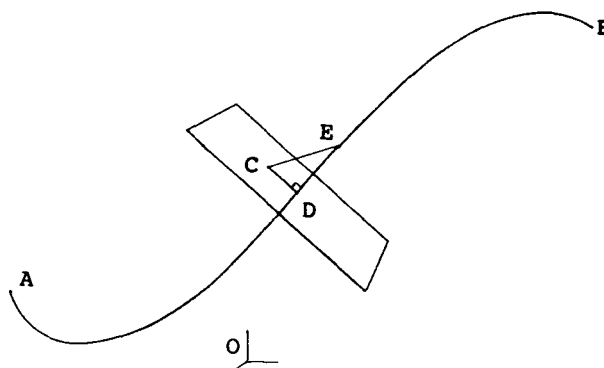


Figure 2 - Local Position Control of Translation

Similarly, the orientation trajectory may be thought of as a displacement along a path on a hypersphere as shown in Figure 3. For a path P on the hypersphere between two orientations A and B, the actual orientation at a particular instant in time t is in error at point C. As before, the next commanded orientation at time $t + \Delta t$ can be found by projecting C normal to the path to point D and then advancing along the path to point E according to the desired time rate of change of orientation. The next orientation can be expressed as [5]

$$\underline{q}(t + \Delta t) = \underline{q}_{E/C} \underline{q}(t) \quad (2)$$

where $\underline{q}(t)$ is the quaternion for orientation C, $\underline{q}(t + \Delta t)$ is the quaternion for orientation E and $\underline{q}_{E/C}$ is the quaternion of the orientation change between E and C.

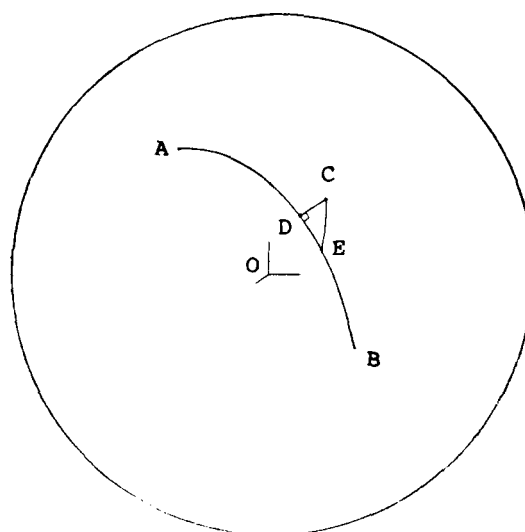


Figure 3 - Local Position Control of Orientation

SOME POSSIBLE REALIZATIONS

The range of possible realizations of local position control encompasses all modes currently described in the literature. Local position control also suggests some new modes not described before. A few of the possible modes are

1. Handcontroller with rate inputs and

- a) Normal and tangential position errors ignored. Equivalent to resolved rate control
- b) Tangential position error ignored.
- c) Neither normal nor tangential errors ignored. Equivalent to resolved rate control with position servo [6].

2. Computer generated path with

- a) One handcontroller rate input for displacement along the path in Δt .
- b) Displacement along the path in Δt is a programmed function of time.

3. Computer generated surface with

- a) Two handcontroller rate inputs for displacement along a path in Δt constrained to the surface.
- b) Displacement along a path in Δt constrained to the surface is programmed function of time.

4. Path generated by a master manipulator driving a slave manipulator

- a) Kinematic control of slave. Unilateral control.
- b) Forces and torques sensed by slave are reflected back to the operator of the master. Bilateral control.
- c) Forces and torques are not sensed by slave but synthesized as a function of the position and orientation difference between master and slave. Synthetic bilateral control.

5. Path generated by a master manipulator driving a slave is constrained to a computer-generated path. Results in the same three cases as mode 4.

6. Path generated by a master manipulator driving a slave is constrained to a computer-generated surface. Results in the same three cases as mode 4.

7. Supervisory level computer generates a path or surface and an operational level computer generates the displacement and orientation rates.

SHUTTLE RMS AS AN APPLICATION

Local position control of the Shuttle RMS would be practical is a means of determining the relative position and orientation of objects in space can be found. Recent work on the Laser Docking Sensor [7] indicates that this may soon be possible with a high degree of accuracy at ranges from 0.1 to 22,000 feet.

One might envision using mode 2 a) with a smooth computer-generated curve superimposed on the TV monitor as a visual aid to the operator. The curve might be heuristically chosen for a particular task like docking or berthing and have perhaps a parametric adjustment. The operator would have to input only a single rate, the rate of displacement along the path, and visually verify that it is following the superimposed curve.

CONCLUSION

Local position control can be considered to be a generalization of the resolved position and rate control concepts. All previously described control modes can be described in the context of local position control and many new ones can be thought of as well. Local position control may well be a big step in the right direction in the evolution of telerobots.

REFERENCES

1. Cohen, A., and Erickson, J. D., "Future Uses of Machine Intelligence and Robotics for the Space Station," Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, pp 200-204.
2. Erickson, J. D., "Manned Spacecraft Automation and Robotics," Proceedings of the IEEE, Vol. 75, No. 3, March 1987, pp 417-426.
3. Sheridan, T. B., "Telerobotics: Computer Aiding for the Human Supervisor," Workshop on Shared Autonomous and Teleoperated Manipulator Control, 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA.
4. Book, W. J., "Teleoperator Arm Design," Handbook of Industrial Robotics, S. Y. Nof, Editor, John Wiley, 1985, pp 138-166.
5. Miller, R. A., "A New Strapdown Attitude Algorithm," Journal of Guidance, Vol. 6, No. 4, July-August 1983, pp 287-291.
6. Kim, W. S., Tendick, F., Ellis, S. R., and Stark, L. W., "A Comparison of Position and Rate Control Telemanipulations with Consideration of Manipulator System Dynamics," IEEE Journal of Robotics and Automation, Vol. RA-3, No. 5, October 1987, pp 426-436.
7. Clowes, T. J., and Schuma, R. F., "Target Acquisition and Track in the Laser Docking Sensor," Society of Photo-Optical Instrumentation Engineers, April 4, 1988.

PRECEDING PAGE BLANK NOT FILMED

DEXTERITY ANALYSIS AND ROBOT HAND DESIGN

Li Lu
Research Associate

A. H. Soni
The L.W.Scott Alter, State of Ohio Eminent Scholar &
Professor of Advanced Manufacturing Systems

Chunsheng Cai
Research Fellow

Max Brown
Professor

Department of Mechanical and Industrial Engineering
University of Cincinnati

ABSTRACT

Understanding about a dexterous robot hand's motion ranges is important to the precision grasping and precision manipulation [4]. This paper presents an object oriented study for a planar robot hand on the ranges, measured with respect to the palm, of position reaching of a point in the grasped object, and of rotation of the object about the reference point. In this paper we introduced the rotational dexterity index and dexterity chart. We developed an analysis procedure for calculating these quantities. We also developed a design procedure for determining the hand kinematic parameters based on a desired partial or complete dexterity chart. These procedures have been tested in detail for a planar robot hand with two 2- or 3-link fingers. We have shown that the derived results are useful to performance evaluation, kinematic parameter design, and grasping motion planning for a planar robot hand.

1. Introduction

Many robot manipulation tasks require a robot hand to have dexterity in fingers' arrangement and movement. These tasks include non-grasping manipulation such as playing piano and pushing object, and grasping manipulation such as power grasping and precision grasping [1, 3, 5, 6]. In this study we are interested in the dexterity for precision grasping. The precision grasping logically implies that the grasped object can move delicately with respect to the palm. Disregarding the grasping capability,

grasping stability, optimal grasping configuration, etc., for which there exist many studies in literature, we further limit ourself to the study of the motion range of the grasped object.

Compared with the robot arm motion, the object's motion relative to hand is usually small. However, for such relative motion the range of position reaching of a point on the object and the range of rotation of the object about the point at a given position is still a basic and important problem. While there are many studies on the robot arm's primary and secondary workspace (see the definition by B. Roth [8] and K.C. Gupta and B. Roth [2]), there are relatively few studies on hand motion range reported in the literature, among which the work by J. Kerr and B. Roth [4] provides us a starting point.

Unlike a robot arm, in conducting motion range for a robot hand, the shape and size of grasped object must be taken into account, since the hand and the grasped object constitute one motion system. [4] has studied the position reaching of a hand with two and three fingers. They assumed a line or a triangle with vertices corresponding to the two or three grasping points. This gives a relatively simple model, however, the influence from the interference between the fingers and the grasped object is neglected to a large extent.

Besides the range of position reaching of a point in the object, the range of rotation of the object about the reference point at each position should also be determined. Since the grasped object usually has very limited range of rotation, it is not

appropriated for one to seek a counterpart in precision grasping for primary workspace which exists for a robot arm. Instead, one need to find the distribution of different rotation ranges in the positionally reachable space.

This paper presents our study guided by above thoughts for a planar robot hand. The paper is divided into two parts: dexterity analysis and design application. In the analysis part, we have selected a circle of suitable size as a testing object, defined a rotational dexterity index for the grasped object at a given position, represented the spatial distribution of rotational dexterity over the workspace by a dexterity chart. Detailed analyses are conducted for a planar hand with two 2- or 3-link fingers. In the design application part, we presented a methodology for designing the kinematic parameters of a planar hand from a desired partial or complete dexterity chart. Depending on the number of controlling points in the dexterity chart, we have shown that the optimal approximate solution in the sense of least square errors can be obtained. The technique of selecting initial value and final solution modification have also been discussed.

2. Dexterity Analysis

We assume in this analysis that the contacts always occur at the finger tips and there is no slipping at the finger tips during motion. For the most part, we also neglect the size of the finger tip. Its influence will be discussed near the end of this section.

In addition to these assumptions, depending on tasks there are various objects with different geometric shapes and sizes. For a particular application, it is appropriate to test directly on the grasped object. In this paper we select a circle as a testing object based on three reasons: (1) without a particular task in mind, a circle is always one of the most simple and common planar objects (or component objects); (2) the simplicity in analysis due to using a circle does not prevent the developed methodology to be extended to other testing objects; (3) the motion range analysis for a circle always provides the lower bound of the motion ranges for any inscribed objects, though this is under the assumption that their contacts are positioned coincident with the circle.

The size of a manipulated object will affect the dexterity measure. The dexterity of a hand decreases as the size of the object increases. In order to study the influence on dexterity due to hand configuration instead of hand dimension, we let the diameter of

the testing circle to be proportional to the average length of the fingers, while the proportional coefficient is chosen heuristically as 0.5. From calculation for many cases in which the hands have two homogeneous 2-link fingers, we noticed that when the proportional coefficient varies around 0.5, a scalar dexterity measure changes linearly.

2.1 Dexterity Analysis of A Planar Hand With Two 2-Link Fingers

When the dimension of the finger tip is not negligible, a planar hand with two 2-link fingers can not turn an object without slipping, since the finger's position and orientation can not be controlled at the same time. To turn an object by pure rolling between the fingers and the object, the minimum number of joints required is two for one finger and three for the other. However, to illustrate our analysis procedure a hand having two homogeneous 2-link fingers with pointed tips is sufficient.

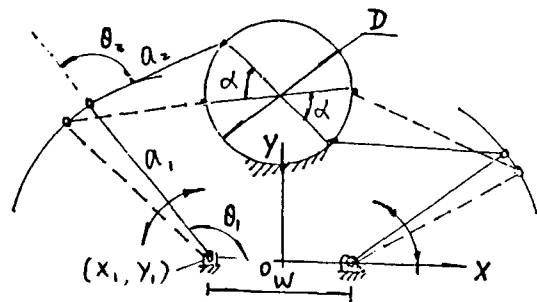


Figure 1

A kinematic model of a hand with two 2-link fingers grasping a ball is shown in figure 1. Since those two fingers are symmetrical, we need to only look at one finger. Let us define, for finger 1, θ_1 to be the rotation angle for joint 1; a_i the length of link i ; (X_1, Y_1) the coordinates of finger base; and (X_b, Y_b) the coordinates for finger tip with respect to palm coordinates. These variable and parameter symbols are consistent throughout this paper.

The transformation matrix from the finger tip coordinates to the palm coordinates, takes a form [7]

$$\begin{bmatrix} M_x & N_x & X_b \\ M_y & N_y & Y_b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & X_1 \\ \sin\theta_1 & \cos\theta_1 & Y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & a_1 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & a_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

..... (1)

ORIGINAL PAGE IS OF POOR QUALITY

We have two equations from (1)

$$(X_b - X_1) \cos \theta_1 + (Y_b - Y_1) \sin \theta_1 = a_2 \cos \theta_2 + a_1$$

..... (2)

$$(Y_b - Y_1) \cos \theta_1 - (X_b - X_1) \sin \theta_1 = a_2 \sin \theta_2$$

..... (3)

The solution of θ_1 and θ_2 should however be checked against any interference between the object and the fingers. If two very near points on the circle can be reached by one finger, their circle-center symmetrical points can be reached by the other finger, and there is no interference between the ball and fingers, we can assume that a corresponding small arc length on the circle is reachable by fingers. Moving the finger around the circle and summing up all reachable small segments, the total reachable arc length can be obtained, which is directly proportional to the range of the

rotation angle of the object, since the contact points are fixed between the fingers and the circle. For a planar precision grasping, we define the Rotational Dexterity Index at a position as

$$RDI = (\text{Sum of reachable Arc Length}) / (2R\pi)$$

..... (4)

The workspace of the hand is defined as the motion range which the reference point of the testing object (a center point of the circle) can reach. Many nodes can be created by dividing the workspace into a collection of small areas. For each node the rotational dexterity index can be calculated. Then, a set of loci of the nodes having equal values of rotational dexterity index can be obtained. Within the range of the workspace, these curves represent the spatial distribution of different degrees of rotational dexterity. We name the area within one curve of rotational dexterity index p as Workspace of Rotational Dexterity Index (WRDI $RDI=p$).

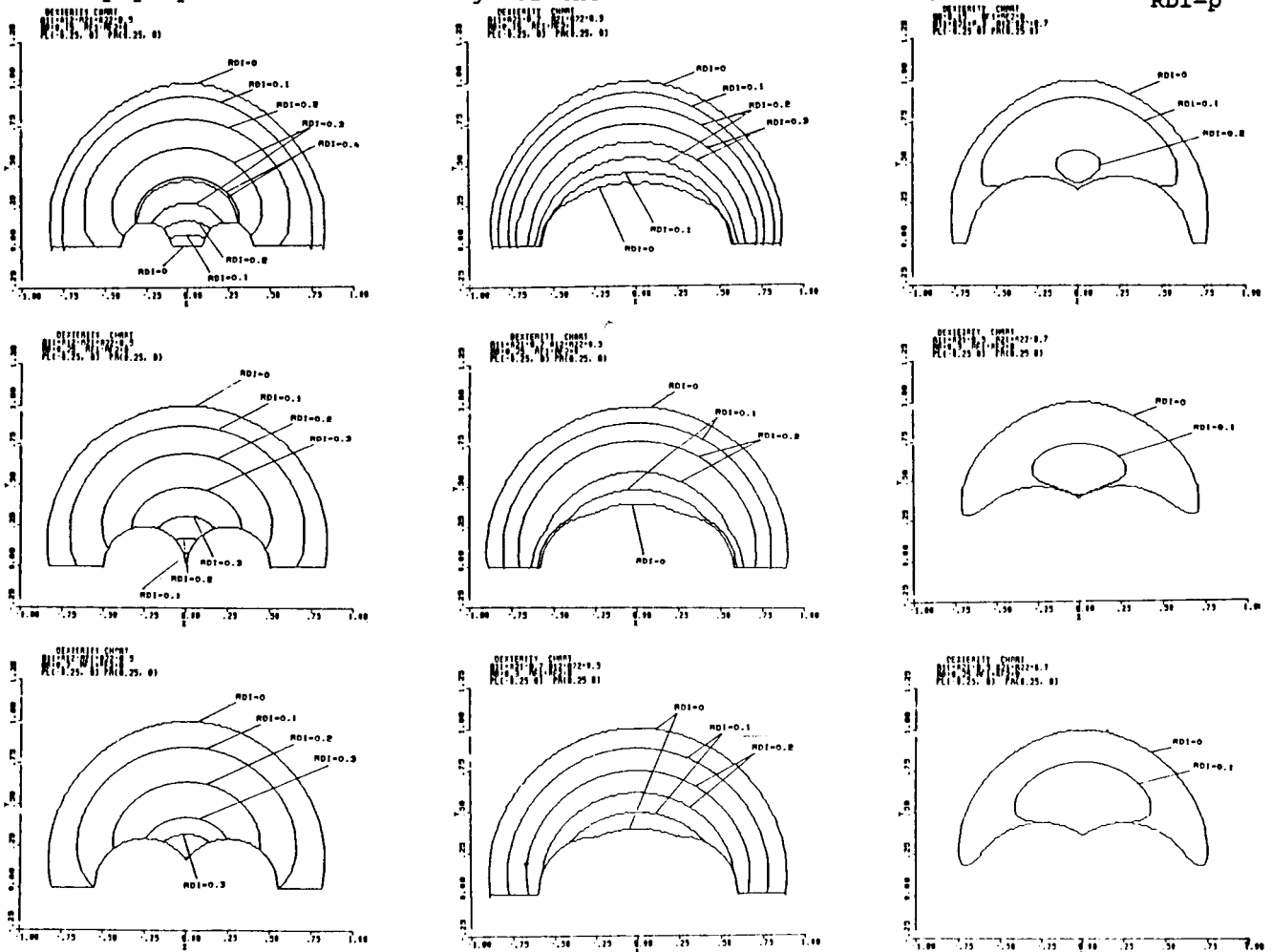


Figure 2

The WRDI distribution can be very effectively expressed through a chart, we call it the dexterity chart.

Figure 2 shows some dexterity charts that present the dexterity of some hands with two homogeneous two-link fingers. In the figures we choose the increase step for RDI as 0.1. From the comparison of the charts we can see clearly that the link parameters have big influence over the manipulating capability of a hand. For a planar hand with two non-homogeneous fingers, a non-symmetrical dexterity chart can be obtained, such as in the figure 3. These dexterity charts are very useful in comparing dexterity of two hands, in programing the motion of a manipulated object, and in designing a hand.

The dexterity of a hand may be expressed by a scalar. We define one scalar as the summing up all the areas in different WRDI. That is,

$$DEX = \sum (i+1) AREA_{RDI=i \cdot step} \quad \dots\dots(5)$$

Where the step can be 0.1 or other value we prefer. The another obvious choice is that define DEX as the average rotational dexterity index.

2.2. Dexterity Analysis of A Planar Hand With Two 3-Link Fingers

The analysis procedure used for the planar hand with two 2-link fingers can be easily extended to a planar hand with two 3-link fingers.

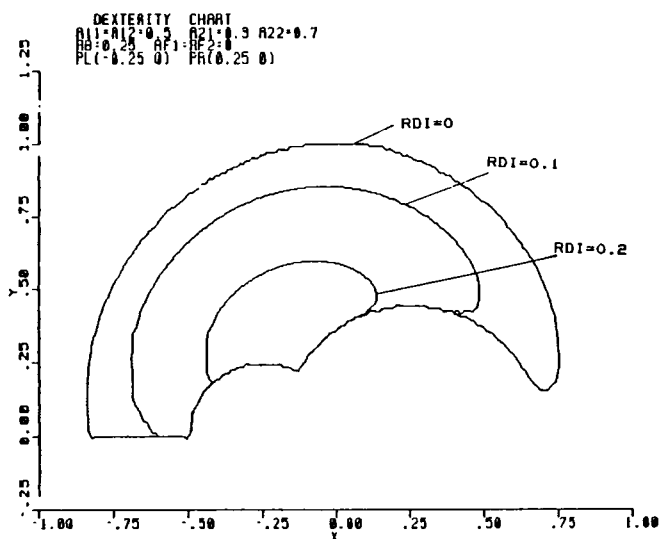


Figure 3

The kinematic model of a two 3-link finger hand is shown in Figure 4. The transformation matrix from finger tip coordinate system to palm coordinate system can be expressed as

$$\begin{bmatrix} M_x & N_x & X_b \\ M_y & N_y & Y_b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & X_1 \\ \sin\theta_1 & \cos\theta_1 & Y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & a_1 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & a_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \dots\dots(6)$$

From (6) we obtain three independent equations

$$\begin{aligned} -M_x a_2 \cos\theta_3 + N_x a_2 \sin\theta_3 + (X_b - M_x a_3 - X_1) &= a_1 \cos\theta_1 \\ -M_y a_2 \cos\theta_3 + N_y a_2 \sin\theta_3 + (Y_b - M_y a_3 - Y_1) &= a_1 \sin\theta_1 \\ \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 &= M_x \cos\theta_3 - M_y \sin\theta_3 \quad \dots\dots(7) \end{aligned}$$

Because the left side of equation (6) consists of known parameters, the variable θ_1 , θ_2 and θ_3 are solvable.

The procedure used for dexterity analysis of a hand with two 2-link fingers can be used now for the analysis of the hand with two 3-link fingers. We need only to follow each steps presented in the previous section.

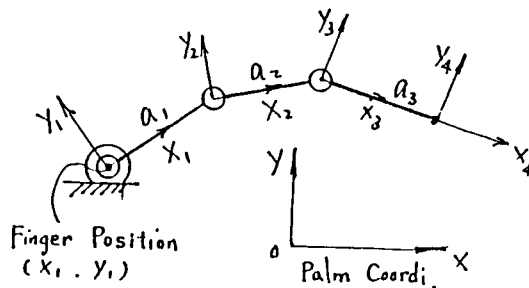


Figure 4

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS OF POOR QUALITY

2.3 The Effect of Finger Tip Dimension on Dexterity

The method can be extended to take into account the dimension of finger tips. Figure 5 shows that when the finger tip is of circular type, we can choose the equivalent radius of the manipulated circle as $R=R+r$, then above procedure for dexterity analysis can still be used. An additional value δRDI should be added to RDI, $\delta RDI = \delta \phi r/R$, where $\delta \phi = \phi_1 - \phi_2$ is due to pure rolling between the finger and the circle.

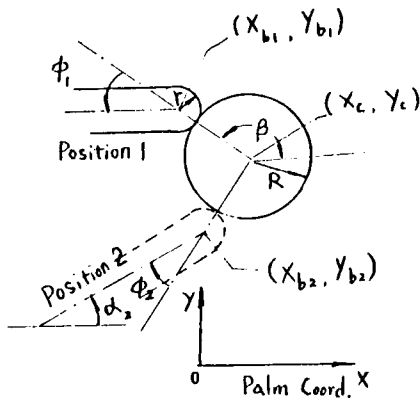


Figure 5

2.4. Dexterity Analysis of A Planar Multifinger Hand

A multifinger hand can reposition some of its finger contacts with respect to the grasped object, so it may have much large WRDI of $RDI=1.0$.

If we assume all fingers keep contacting while turning a circle, the previous methodology can be applied directly.

3. Application for Kinematic Parameters Design of Planar Robot Hands

Given the number of fingers, numbers of links for each finger, and finger tip positions, there are still the kinematic parameters of a planar hand need to be determined. These parameters include palm size (or locations of finger bases), the length of each finger link, and the diameters of the sphere (semisphere) attached on finger tips.

We present here a design methodology which is based on a partial or complete desired dexterity chart. Such partial dexterity chart can be obtained by selecting several positions on the boundaries of different WRDI. For an arbitrarily given object and a dexterity chart, while we can design a hand to follow accurately few

controlling points in a dexterity chart, it is not feasible to have a hand whose dexterity chart matches exactly an arbitrarily given one. We develop an approximate design method which results in a hand with the dexterity chart that has minimum deviation from a given one.

3.1 Design of A Planar Hand with Two 2-Link Fingers

The configuration of a planar two-link finger is the same as shown in figure 1. As we illustrated in figure 1 and figure 5, let (X_1, Y_1) be the coordinates for the base of finger 1 in the palm coordinate; (X_c, Y_c) be the coordinates for the center point of the circle; β be the angle between the X axis of the palm frame and the line through the circle center and contact point of finger1; and r be the radius of the sphere attached on the tip of finger 1. The transformation matrix from finger tip coordinate system to the palm coordinate system takes the same expression as in (1). Except that $X_b = X_c + (R+r)\cos\beta$ and $Y_b = Y_c + (R+r)\sin\beta$. Premultiply both sides of the equation by the inverse matrix associated with θ_1 . Three independent equations are obtained.

$$a_2 \cos \theta_2 + a_1 = \cos \theta_1 [X_c + (R+r) \cos \beta - X_1] + \sin \theta_1 [Y_c + (R+r) \sin \beta - Y_1] \quad \dots (8)$$

$$a_2 \sin \theta_2 = -\sin \theta_1 [X_c + (R+r) \cos \beta - X_1] + \cos \theta_1 [Y_c + (R+r) \sin \beta - Y_1] \quad \dots (9)$$

$$\cos \theta_2 = M_x \cos \theta_1 + M_y \sin \theta_1 \quad \dots (10)$$

Because $M_x = \cos \alpha$ and $M_y = \sin \alpha$, where α is the angle between last link of the designed finger and the X axis of the palm frame, the equation (10) can be rewritten as

$$\alpha = \theta_1 + \theta_2 \quad \dots (11)$$

Given one position in a dexterity chart, there are three equations and six unknowns $(X_1, Y_1, a_1, a_2, \theta_1, \text{ and } \theta_2)$. For n positions in the dexterity chart, there are $3n$ equations and $4+2n$ unknowns. Therefore, for 4 given positions in a dexterity chart there exists an exact solution for finger parameters.

When the number of given positions are over 4, the above equations are overconstrained. we can obtain approximate solutions by using regression methods, in which one of the simplest is least square method. Our objective is to seek suitable $a_1, a_2, X_1, Y_1, \theta_1$, and θ_2 , to minimize the error for equations (8), (9) and (11).

$$\begin{aligned} \min \Sigma \{ & \cos\theta_{1i}[X_{ci}+(R+r)\cos\beta_i-X_1]+ \\ & +\sin\theta_{1i}[Y_{ci}+(R+r)\sin\beta_i-Y_1]-a_2\cos\theta_{2i}- \\ & -a_1 \}^2 + \{ -\sin\theta_{1i}[X_{ci}+(R+r)\cos\beta_i-X_1]+ \\ & +\cos\theta_{1i}[Y_{ci}+(R+r)\sin\beta_i-Y_1]-a_2\sin\theta_{2i} \}^2 \\ \text{S.T. } & \theta_{1i} + \theta_{2i} - \alpha_i = 0 \\ \text{where } & i = 1, 2, 3, \dots, n \end{aligned} \quad \dots (12)$$

In using least square method, we tentatively guess θ_{2i} for each position, and solve θ_{1i} from equation (11). For n positions the $2n$ equations from (8) and (9) can be written as in matrix form

$$\begin{bmatrix} 1 & K_{12} & K_{13} & K_{14} \\ 0 & K_{22} & K_{23} & K_{24} \\ 1 & K_{32} & K_{33} & K_{34} \\ 0 & K_{42} & K_{43} & K_{44} \\ \dots & & & \\ 1 & K_{(2n-1)2} \dots K_{(2n-1)4} \\ 0 & K_{(2n)2} \dots K_{(2n)4} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \\ \dots \\ G_n \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ \dots \\ P_{2n} \end{bmatrix} \quad \dots (13)$$

where $G_1=a_1$, $G_2=a_2$, $G_3=X_1$, $G_4=Y_1$

$$\begin{aligned} P_{2i-1} &= \cos\theta_{1i}[X_{ci}+(R+r)\cos\beta_i]+ \\ & +\sin\theta_{1i}[Y_{ci}+(R+r)\sin\beta_i] \\ P_{(2i)} &= -\sin\theta_{1i}[X_{ci}+(R+r)\cos\beta_i]+ \\ & +\cos\theta_{1i}[Y_{ci}+(R+r)\sin\beta_i] \\ K_{(2i-1)2} &= \cos\theta_{2i} \\ K_{(2i)2} &= \sin\theta_{2i} \\ K_{(2i-1)3} &= K_{(2i)4} = \cos(\alpha_i-\theta_{2i}) \\ K_{(2i-1)4} &= -K_{(2i)3} = \sin(\alpha_i-\theta_{2i}) \\ i &= 1, 2, 3, \dots, n. \end{aligned}$$

Rewriting equation (13) in brief form

$$[K]_{(2n \times 4)} [G]_{(4 \times 1)} = [P]_{(2n \times 1)} \quad \dots (14)$$

Premultiplying both sides of equation (14) by matrix $[K]_{(4 \times 2n)}^T$,

$$[G] = ([K]^T [K])^{-1} [K]^T [P] \quad \dots (15)$$

The matrix $[K]^T [K]$ is positive definite and symmetrical. Except for singular cases it has inverse matrix.

3.2 Initial Values of Iteration

From kinematic parameters obtained by (15), actual θ_{1i} and θ_{2i} can be calculated. By iterative computation the solution will be refined each time.

The initial values of θ_{2i} 's will affect the convergence speed. In the design of a two 2-link finger, by our sign convention, negative initial values are suggested for a left finger, and positive initial values are suggested for a right finger.

Looking at the matrix expression

$$[K]^T [K] = \begin{bmatrix} n & \Sigma \cos\theta_{2i} \\ \Sigma \cos\theta_{2i} & n \\ \Sigma \cos(\alpha_i-\theta_{2i}) & \Sigma \cos\alpha_i \\ \Sigma \sin(\alpha_i-\theta_{2i}) & \Sigma \sin\alpha_i \\ & \Sigma \cos(\alpha_i-\theta_{2i}) & \Sigma \sin(\alpha_i-\theta_{2i}) \\ & \Sigma \cos\alpha_i & \Sigma \sin\alpha_i \\ & n & 0 \\ & 0 & n \end{bmatrix}$$

we notice that to avoid singularity we can not have all initial values of θ_{2i} 's to be zero, or to satisfy the constraints $\alpha-\theta_2=0$ or $\alpha-\theta_2=\pi/2$ for all positions.

3.3 Approaching Final Solution

The solution from the iterative method means that the finger tip can be as close as possible to the required positions for given α_i 's. It is not surprise that some part of the dexterity chart will be satisfactory and some part will be not. Following three techniques can help us to improve the final solution, while using the same number of links for each finger and same number of finger for each hand.

(1) Improve α 's

From (13) we know that we can not assign all α_i equal 0 or $\pi/2$, otherwise the matrix $[K]^T [K]$ will be singular. Checking the result of computation, we can find which α is selected unappropriately and to which direction it needs to be modified.

(2) Repeat The Equations for Few Positions In (13)

Repeating the equations for few positions in (13) would drive the corresponding parts of the dexterity chart approach more closely to desired one, though the other parts will deviate correspondingly. This is useful for the case where some points are relatively more important.

(3) Shift The Given Positions of The Finger Tip

ORIGINAL PAGE IS OF POOR QUALITY

The resulted dexterity chart may also provide us some clues as to shift some contact positions of the finger tip, since either their requirement can not be met, or they have big negative influence over whole chart.

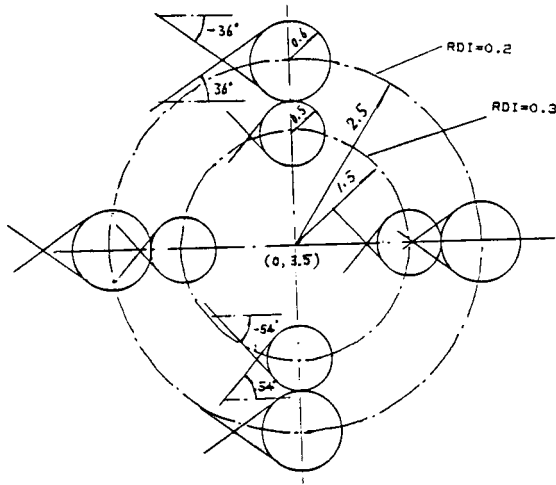
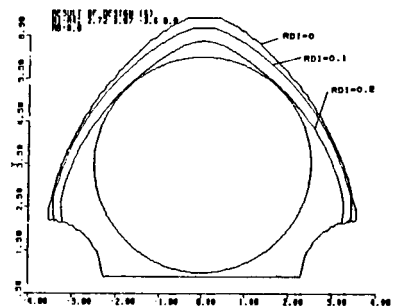
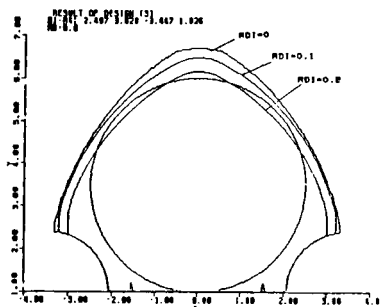
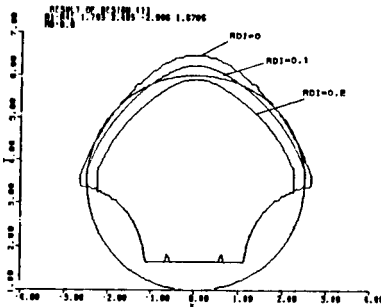


Figure 6

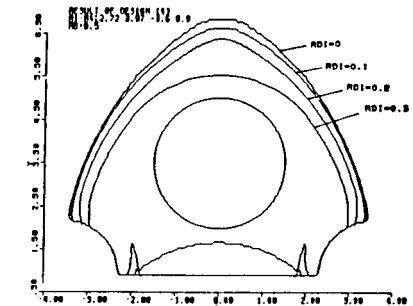
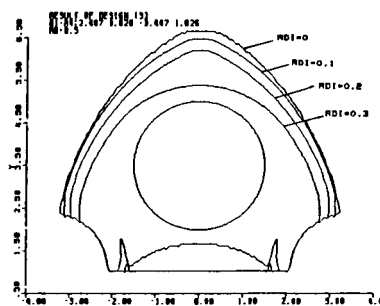
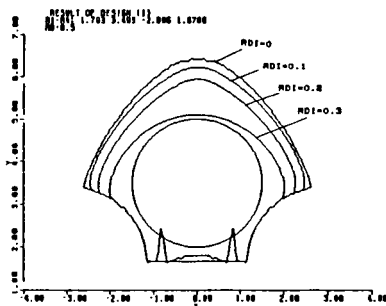
3.4 An Design Example

A hand with two 2-link fingers is designed to test our design methodology. The hand is expected to (1) have a WRDI ($RDI=0.2$) in a circle with radius $R=2.5$, for a manipulated ball with radius $R=0.6$; and at the same time (2) have a WRDI ($RDI=0.3$) in a circle with radius $R=1.5$, for a manipulated ball with radius $R=0.5$. Both of the circle centers are at the point $(0, 3.5)$. Totally 16 positions in the dexterity chart are selected to design the hand. Because the desired dexterity chart and fingers' configuration is symmetric, only one finger need to be designed. The selection of initial positions and orientations of the last links is shown in figure 6. The process of approaching the final solution is shown in figure 7. To improve the legibility, RDI curves with larger values are not shown in the charts. Figure 7 (b) shows that the condition (2) is always satisfied, and figure 7 (a) shows how the condition (1) is approached.

Although, in the example, two manipulated objects are circles, the



(a)



(b)

Figure 7

method can be used to design a hand for manipulating several different objects with several different levels of dexterity. That is very useful for the designing of industrial robot hands.

3.5 Design of a Three Link Finger

The design method can be extended to design a three link finger.

The configuration of a planar three link finger is figure 4. The transformation matrix from finger tip coordinate system to the palm coordinate system is expressed in (6).

Considering $M_x = N_y = \cos \alpha$, $M_y = -N_x = \sin \alpha$, rewrite equation (7), we have

$$\alpha = \theta_1 + \theta_2 + \theta_3 \quad \dots\dots(16)$$

$$a_1 \cos \theta_1 + a_2 \cos(\alpha - \theta_3) + a_3 \cos \alpha + X_1 = X_b \quad \dots(17)$$

$$a_1 \sin \theta_1 + a_2 \sin(\alpha - \theta_3) + a_3 \sin \alpha + Y_1 = Y_b \quad \dots(18)$$

When the dimension of the finger tip is considered, $X_b = X_c + (R+r) \cos \beta$ and $Y_b = Y_c + (R+r) \sin \beta$ are to be substituted into (17) and (18).

For one point on a dexterity chart there are 3 equations and 8 unknowns ($X_1, Y_1, a_1, a_2, a_3, \theta_1, \theta_2$

and θ_3). For n points there are 3n equations and 5+3n unknowns. After we assign values to 5 unknowns, we can have 3n equations with 3n remain unknowns. It is very difficult or impossible to have analytic solution, so we resort to using numerical method.

In using the least square method we have to assign two θ 's for each position, and the third θ can be solved from equation (16). Equations (17) and (18) can be used to construct a matrix equation

$$[K]_{(m \times 5)} [G]_{(5 \times 1)} = [P]_{(m \times 1)} \quad \dots(19)$$

where $m=2n$

$$G_1=a_1, G_2=a_2, G_3=a_3, G_4=X_1, G_5=Y_1$$

$$K_{(2i-1)1} = C^{\theta_{1i}}$$

$$K_{(2i-1)2} = \cos(\alpha_i - \theta_{3i})$$

$$K_{(2i-1)3} = \cos \alpha_i$$

$$K_{(2i)1} = S^{\theta_{1i}}$$

$$K_{(2i)2} = \sin(\alpha_i - \theta_{3i})$$

$$K_{(2i)3} = \sin \alpha_i$$

$$K_{(2i-1)4} = K_{(2i)5} = 1$$

$$K_{(2i-1)5} = K_{(2i)4} = 0$$

The matrix $[G]$ can be expressed as

$$[G] = ([K]^T [K])^{-1} [K]^T [P] \quad \dots\dots(20)$$

The rest of the steps are similar as presented for designing a 2-link finger.

3.6 Other Issues On Design

1. Design of Multifinger Hands

In the above procedure a hand is designed with one finger at a time, so there is no difference between designing a multifinger hand and designing a two finger hand. However, the chance of interference will increase as the finger number increase.

2. Selection of The Size of Finger Tip

Theoretically, we can design the diameter of a circular finger tip using the equation (3). However, sometime the result is not feasible. The better way is we first select the diameter of the finger tip according to the size of the manipulated object, Then check whether the result is satisfactory.

4. Conclusion

Inspired by the work on dexterous robot hand workspace [4], we proceeded an object oriented study for a planar hand on the range of position reaching of a point on the grasped object, and the range of rotation of the object about the reference point. Both ranges are measured with respect to the palm.

In this paper, for each position of the reference point, the percentage of one revolution that object can rotate about the reference point with respect to the palm is defined as rotational dexterity index for that point. The distribution of such rotational dexterity index over the whole positionally reachable space is expressed in a dexterity chart, which includes contours of different index values.

In addition to using dexterity chart for hand performance evaluation, we also presented an design procedure for determining the kinematic parameters of a planar hand with revolute joints based on a desired partial or complete dexterity chart. Least square error iterative method, its initial value selection and final solution improvement have been stressed.

The dexterity analysis and design application are presented in a case study format, in which a planar hand with two 2- or 3-link fingers have been investigated in detail.

Although the methodologies are developed for dexterity analysis and

design of planar hands, they can be extended to the spatial hands for the same purpose.

The methods are useful in the analysis, design and motion planning of industrial robot hand for assembly tasks or some other tasks in which precision grasping and manipulation are needed.

- [10] Yang, D.C.H. and Lai, Z.C., "On the Dexterity of Robotic Manipulators -- service angle", pp.262-270, Journal of Mechanism, Transmissions, and Automation in Design, Vol.107, June 1985

References

- [1] Cutkosky, Mark R. and Wright, Paul K., "Modeling Manufacturing Grips and Correlations with the Design of Robotic Hands", pp.1533-1539, Proceeding of IEEE International Conference on Robotics and Automation, 1986
- [2] Gupta, K.C. and Roth, B., "Design Considerations for Manipulator Workspace", pp.704-711, Transactions of The ASME, Vol. 104, October, 1982
- [3] Jacobsen, S.C. et al, "Design of The UTAH/M.I.T. Dexterous Hand", pp. 1520-1532, Proceeding of IEEE International Conference on Robotics and Automation, 1985
- [4] Kerr, Jeffrey R., and Roth, B. "Analysis of Multifingered Hands", pp. 3-17, The International Journal of Robotics Research, Vol. 4, No.4, Winter 1986
- [5] Lyons, Damian M., "A Simple Set of Grasps for a Dexterous Hand" pp. 588-593, Proceeding of IEEE International Conference on Robotics and Automation, 1985
- [6] Mason, Matthew T. and Salisbury, J.K., Robot Hands and the Mechanics of Manipulation, The MIT Press, 1985
- [7] Paul, Richard P., Robot Manipulators: Mathematics, Programming, and Control, The MIT Press, 1981
- [8] Roth, B., "Performance Evaluation of Manipulators from A Kinematic viewpoint", NBS Special Publication Performance Evaluation of Programable Robots and Manipulators, 1975, pp.39-61
- [9] Vijaykumar, R., Waldron, K.J., Tsai, M.J., "Geometric Optimization of Serial Chain Manipulator Structures for Working Volume and Dexterity", pp.91-103, The International Journal of Robotics Research, Vol. 5, No.2, Summer 1986

ORIGINAL PAGE IS
OF POOR QUALITY

CONCEPT FOR A

LARGE MASTER/SLAVE-CONTROLLED ROBOTIC HAND

William A. Grissom, Ph.D., P.E.
Mahmoud A. Abd-Allah, Ph.D.
Manufacturing Engineering Department
Central State University
Wilberforce, Ohio 45384

Carl L. White
Computerized Technologies, Inc.
1445 Summit Street
Columbus, Ohio 43201

ABSTRACT

A strategy is presented for the design and construction of a large master/slave-controlled, five-finger robotic hand. Each of the five fingers will possess four independent axes each driven by a brushless DC servomotor and, thus, four degrees-of-freedom. It is proposed that commercially available components be utilized as much as possible to fabricate a working laboratory model of the device with an anticipated overall length of two-to-four feet (0.6 to 1.2 m). The fingers are to be designed so that proximity, tactile, or force/torque sensors can be imbedded in their structure. In order to provide for the simultaneous control of the twenty independent hand joints, a multilevel master/slave control strategy is proposed in which the operator wears a specially instrumented glove which produces control signals corresponding to the finger configurations and which is capable of conveying sensor feedback signals to the operator. Two dexterous hand master devices are currently commercially available for this application with both undergoing continuing development. A third approach to be investigated for the master control mode is the use of real-time image processing of a specially patterned master glove to provide the respective control signals for positioning the multiple finger joints.

INTRODUCTION

Objective

It is proposed to design and construct a large, anthropomorphic, master/slave-controlled, robotic hand. The envisioned device would be larger and more powerful than the human hand while possessing sufficient dexterity to closely mimic the fingering and grasping configurations of its human counterpart. The device has been assigned the

acronym, SLAVE² for, "Servomotor-Linked Articulated Versatile End Effector," reflecting the planned master/slave control mode and the use of an individual electric servomotor to drive each joint.

A rapid prototype R&D strategy utilizing off-the-shelf components wherever possible is proposed for the development of the SLAVE² laboratory prototype. A key goal of the strategy is to minimize development time and costs by eliminating long lead times for design and construction of individual components. The commercial availability of components including the electric servomotors and power transmission mechanisms to drive the individual finger joints will, thus, dictate size, weight, payload and finger length of the hand assembly. Based upon this consideration, it is anticipated that the initial working laboratory model will have an overall length of two-to-four feet (0.6-1.2 m) and an individual finger clamping force of 15-20 pounds (.067-.089 kN).

Previous Work in Robotics Hands and Grippers

Classification of robot hands may be based on mechanical characteristics. The basic criteria for classification is the number of "degrees of freedom" which the robot hand possesses. Degrees of freedom relates to the number of powered joints and the kinematics of the hand. The kinematics plus the geometry of the hand determine the envelope of the work space. Theoretically, the larger the number of degrees of freedom the more dexterous the hand and the more numerous the grasping patterns which can be achieved.

The current research in robotic hands has been motivated by extensive work in prosthetics and industrial grippers. In industry, grippers are designed for

securely grasping objects and the burden of maneuvering them is put on the robot joints where these grippers are attached. For various objects, different grippers with different finger shapes and actuation mechanisms are designed to be simple and durable [1]. For positioning small objects, industrial robots have been provided with small motion hands. An elegant example, is the Tokyo hand where three five-bar structures achieve 6-degrees of freedom to maneuver a triangular plate [2].

To improve the dexterity of the robot hands, computer controllers have been incorporated. These computerized controllers provide the potential of designing articulated robot hands with programmable and reconfigurable fingers for different applications.

In designing prosthetic devices, much of the work has focused on developing simple grasping systems. An example of a simple single degree-of-freedom prosthetic hand is the Utah hand where myoelectric sensing has been employed [3]. A more advanced prosthetic device employing force balancing mechanics to allow fingers to curl and settle around the grasped object is the Belgrade hand [4].

Dexterity is a key feature in designing robot hands. Thus multi-degree-of-freedom hand designs have been reported in recent years. These designs employ a wide range of actuating devices (e.g. pneumatic, hydraulic and electric servomotors) and force and tactile sensing. In particular, the dexterous robot hand developed at Japan's Electro-Technical Laboratory is a computer controlled 3-finger, 11-degree-of-freedom device able to grasp and impart controlled motions to rectangular and spherical objects [5].

Another remarkable design was the Stanford/JPL hand which has three, 3-degree-of-freedom fingers [6]. That design proved that three fingers are sufficient to grasp an object assuming that the object is held in the fingertips. That grasping technique is not very secure since efficient grasping requires that fingers curl around

objects and hold them against the palm. Also developed was a hierarchical control system for commanding the fingers and object motions [7].

Using the shape memory alloy (SMA) technology, Hitachi has combined the actuator and transmission into one mechanical element [8]. The Hitachi hand is kinematically similar to the Tokyo hand. The slow response time, high power consumption, and actuator fatigue failure are three significant problems with this hand design.

The most complex mechanical hand is the UTAH/MIT hand [9-11]. The current design has a total of 17 degrees of freedom and consists of three, 4-degree-of-freedom fingers and one, 4-degree-of-freedom thumb. The hand presents a major success in designing a very complex mechanical system and in the high performance achieved. The major drawback is the high power consumption and the large amount of computation resources (3-5 Motorola 68000 microprocessors).

Common to all of these systems are a number of major design issues. The number of digits affect the effectiveness of secure grasping while more joints broaden the range of grasping capabilities. One of the major issues in designing dexterous robot hands is the choice of actuation and transmission mechanisms. To avoid excessive weight, actuators are often located away from the hand and a variety of transmission schemes are utilized. In particular, cables routed through flexible conduits and Kevlar composite flat ribbons passing over pulleys are used.

As indicated, dexterous hand designs are characterized by the kinematics, actuators, and transmission mechanisms. In the following section, these key factors will be discussed briefly for the proposed SLAVE² dexterous hand.

PROPOSED ROBOTIC HAND DESIGN

Kinematics

A mechanical hand configuration possessing four fingers and a thumb is

ORIGINAL PAGE IS OF POOR QUALITY

contemplated. Each of these five members will have four joints or degrees-of-freedom. More specifically, for each finger/thumb member three joints would provide flexion and extension (and possibly hyperextension) and a fourth joint would allow abduction and adduction. This would give the hand a total of twenty degrees of freedom and provide sufficient dexterity to closely replicate the gripping and fingering actions of a human hand.

Actuators

Each of the twenty joints is to be directly driven by an independent DC servomotor and integrated speed reducing mechanism. The brushless DC type of servomotor duplicates the external performance of a conventional DC motor without utilizing a commutator or brushes. This is possible because solid-state electronic switching replaces the conventional brush commutation switching process. A second major difference is that the wound member, or armature, reverses its role and relative position from rotor (rotating member) and inner component in the conventional DC motor, to stator (stationary member) and outer component in the brushless motor. These two differences lead to a number of significant advantages from the brushless DC motor [12-13]:

1. No brushes to wear out: increased reliability, reduced maintenance requirements.
2. No commutator bars to oxidize: ability to sit idle for years without loss of performance.
3. Absence of brush arcing: safer in the presence of fumes, dust, paint spray, etc.
4. Speeds up to 80,000 RPM are practical.
5. Less radio-frequency interference.
6. Easier cooling of windings with fins or cooling jacket: extended operating range.
7. Smaller diameter, more compact.
8. Reduced inertia: increased acceleration and improved control.

Although practical brushless DC servomotors are a relatively recent development triggered by advances in

solid state electronics and permanent magnet technology, units are now available from a number of major manufacturers. Included in this category are suppliers such as Inland, Moog, Litton Clifton Precision, Fanuc, Indramat, Mavilor and Ejectorcraft.

Power Transmission

Electric motors characteristically produce relatively low torque in the low speeds range. This is true as well for brushless DC motors, and preliminary calculations indicate that torque multiplication (or speed reduction) rates in the area of 200:1 will be required to achieve the desired robotic hand strength. To meet this requirement, the patented harmonic drive gearing device available from the Harmonic Drive Division of the Emhart Machinery Group, Wakefield, MA, has been tentatively identified. The harmonic drive has three simple, concentric components:

- 1) a rigid circular spline with internal gear teeth, the outermost component which is a non-rotating member for speed reducing applications;
- 2) a non-rigid "flexspline" with external gear teeth, the intermediate member of the assembly serving as the output member for speed reducing applications;
- 3) the elliptical wave generator, the innermost of the concentric components serving as the input member with its elliptically shaped inner bearing race and ball bearings rotating within an outer bearing race of the "flexspline" output member.

The fact that the outer rigid circular spline has two more teeth than the mating "flexspline" results in a relative angular motion between these two components equal to the spacing of two teeth for each complete rotation of the elliptical wave generator. With the circular spline rotationally fixed, the "Flexspline" will rotate in the opposite direction to the input at a reduction ratio equal to the number of teeth on the "Flexspline" divided by two. A detailed explanation of these operating principles is given in the "Harmonic

Drive Designers Handbook" [14] along with load and accuracy ratings, operating life expectancies and installation and servicing guidelines,

The unique design of the harmonic drive yields the following advantages for robotics applications:

1. Exceptionally high torque and power capability in a small package.
2. Essentially zero backlash.
3. Efficiencies as high as 90%.
4. Ratios as high as 320:1 in a single reduction with much higher ratios achieved by compound stages.
5. Concentric input and output shafts.
6. No radial loads since torque is generated by a pure couple; this simplifies the supporting structure requirements.

Drawbacks of the harmonic drive are that it is relatively compliant exhibiting a soft windup characteristic in the low torque region, and that it produces a small, sinusoidal positional error on the output. This error varies inversely with the pitch diameter at a predominant frequency of twice the input speed. Additionally an amplitude modulation typically occurs twice per output revolution.

Electronic Programmable Controllers

With the many degrees of freedom required for dexterous robot hands, the problem of control and demand on computing escalates. The simplest approach is to use a local control loop for each joint. However, for precise motion control, a coordinated motion for fingers and digits becomes a must for an efficient design.

For master-slave operation, where the coordination is achieved by the action of a human in the loop, the coupling between the fingers is neglected. Currently, a number of high performance servomotor controllers are commercially available. These controllers are designed to be programmable and installed in personal computers.

DEXTEROUS HAND MASTERS

The proposed master/slave control mode

calls for the operator to wear a specially instrumented dexterous hand master. This device must produce control signals capable of directing the servomotor actuators of the robotic slave hand into correspondence with the respective positions of the human operator's hand joints. Plans call for consideration of three different dexterous hand masters to carry out this control function. Two such devices, the A. D. Little "Sarcos Dexterous Hand Master" and the VPL Research "DataGlove" are currently commercially available though both are undergoing continuing development. Their application will be discussed briefly in the following paragraphs together with a third approach utilizing real-time image processing of a special optically patterned master glove.

A. D. Little Sarcos Dexterous Hand Master

Arthur D. Little, Inc. of Cambridge, Massachusetts [15] offers a Sarcos Dexterous Hand Master. The device utilizes mechanical linkage assemblies secured to the individual finger digits by means of flexible ring-like bands. Built-in hall effect potentiometers translate the various linkage motions into electrical signals which can be correlated to the individual finger joint movements.

The linkages are constructed chiefly of aluminum, non-magnetic stainless steel, and delrin. A stainless steel hand clip is used to fasten the device to the hand. Special provisions include spring loading of the ring-like finger bands to maintain proper positioning even though the fatty tissue of the finger changes shape as the joints flex; and, "passive pivots" perpendicular to the joint bending axes to accommodate the non-parallel joint axes commonly resulting from crooked fingers.

Currently, up to twenty human joints motions can be monitored with a resolution of one-half degree over their full range for flexion or ab/adduction. Each channel is sampled 100 times per second to provide for real time finger configuration data. Accuracy of

ORIGINAL PAGE IS OF POOR QUALITY

positioning and repeatability are said to be strong points of the A. D. Little hand master.

VPL DataGlove

VPL Research of Redwood City, California markets the DataGlove [16-18], an ingenious glove-like dexterous hand master that senses hand gesture position and orientation in real time. The device utilizes fiber-optic cables sandwiched between a stretchable inner glove and a cloth outer glove.

Each joint motion to be detected requires a separate fiber-optic cable laid in a parallel path running across the joint and looping back so that both free ends are anchored in an interface board mounted near the wrist. At one end of the cable is a light emitting diode source and at the other a phototransistor. The segments of the cable which rest over the joint are specially treated so that the light escapes when the joint is flexed. The greater the degree of bending, the greater is the loss of transmitted light. This effect can be detected by the phototransistor and calibrated to provide angular measurements with a resolution of one degree. A data acquisition rate of 60 times per second is used.

An additional feature on the DataGlove Model 2 System is a high resolution, 3D, magnetic digitizing device which provides for 6-degree-of-freedom (three translation and three rotation coordinates) tracking of the absolute position of the hand. This tracking device, produced by Polhemus Navigation Sciences Division of the McDonnell Douglas Electronics Co., is designated as the "3SPACE Isotrak."

It should be noted that VPL Research has recently developed a counterpart of the DataGlove hand master called the DataSuit which provides configuration data for the entire body.

Optical Pattern Hand Master

A third method suggested for the master control mode is the use of a master glove imprinted with a special color-

coded optical pattern. In this approach, the respective control signals for positioning the multiple finger joints would be extracted from the glove image. Potentially, the required glove could be lighter, better fitting, less cumbersome, and less expensive than either the A. D. Little Sarcos Dexterous Hand Master or VPL DataGlove. The authors are not aware of any commercially available devices of this nature or any researchers who have applied this approach to date.

Nevertheless, the continuing gains in computing speed which are being achieved by parallel processing may make this a viable approach for mimicking the relatively slow, four-to-five hertz, action of the human hand. To achieve real time performance, however, while compensating for factors such as extraneous light or the masking effect of hidden fingers, special treatments might be required. For example, consideration might be given to the application of neural networks by which the system could be trained to quickly recognize specific finger configurations, or to the use of special image enhancing techniques such as the eigenimaging technique [19-20].

APPLICATIONS

The proposed SLAVE² approach has attracted preliminary support from the National Aeronautics and Space Administration (NASA). The device would serve as a laboratory model to develop end effector technology for Space Shuttle servicing and Space Station construction, servicing and repair operations. Tentative plans call for the construction of two similar models, one for study by the Kennedy Space Center (KSC), and the second for the Jet Propulsion Laboratory (JPL).

The KSC SLAVE² hand will be designed for installation on the IRB-90/2 Anthropomorphic Robot manufactured by ARI. The IRB-90 is capable of lifting approximately 200 pounds (90 kg) and holding it about ten feet (3000 mm) from its base with a repeatability of 0.04 inches (1 mm) under constant operational conditions. Further details of this system installed at the Robotics

Applications Development Laboratory (RADL) have been reported by V. L. Davis [21] of KSC.

The JPL model of the SLAVE² hand will be designed to attach to a laboratory work stand for testing and evaluation.

Anticipated commercial applications include handling of hazardous wastes, munitions, or large radioactive or chemically contaminated objects. Fire fighting, construction, demolition, disaster clean-up, and rescue operations might provide additional applications for a large dexterous end effector operated remotely under master/slave control.

ACKNOWLEDGEMENTS

Support to initiate the proposed SLAVE² development has been provided by NASA Headquarters, Kennedy Space Center, the Jet Propulsion Laboratory, and the State of Ohio. Funding is being provided by NASA Grant No. NAGW 1336, Jet Propulsion Laboratory Contract No. 958292, and an Ohio Board of Regents Research Challenge grant. Mr. James Aliberti of Kennedy Space Center and Dr. Edwin P. Kan of Jet Propulsion Laboratory have agreed to serve as technical monitors.

REFERENCES

1. Lundstrom, B., "Industrial Robotics - Gripper Review," International Fluidics services, Ltd., Bedford, England, 1977.
2. Inoue, 3rd International Symposium on Robotics Research (ISSR), Oct. 1985, Gouvieux, France. Pub. by MIT Press, 1985.
3. Jacobsen, S. Knutti, D.; Johnson, R.; Sears, H. "Development of the Utah Artificial Arm," IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Vol. BME-29, No. 4, April 1982, 0018-9294/82/0400-0249, 1982.
4. Rakic, M. "Belgrade Hand Prosthesis," Symposium on Basic Problems of Prehension Movement and Control of Artificial Limbs, London, 1968.
5. Okada, T. "Computer Control of Multijointed Finger System for Precise Object Handling, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Vol SMC-12, No. 3 May/June 1982.
6. Salisbury, J. "Kinematic and Force Analysis for Articulated Hands," Ph.D. dissertation in mechanical engineering, Stanford Computer Science Dept. Report # STAN-CS-82-921, June 1982.
7. Salisbury, J. et al, "Integrated Language, Sensing and Control a for Robot Hand," Proceedings of the 3rd ISRR, Gouvieux, France, Oct. 1985.
8. Hitachi, "Hitachi's SMA Robot Hand," Press release from Hitachi Mechanical Engineering Research Labs, Ibaraki, 300, Japan, 1983.
9. Jacobsen, S. Knutti, et al, "The UTAH/MIT Dexterous Hand," Work in Progress, Center Biomed. Des., Dept. of Mech. & Indus. Eng., Univ of Utah.
10. Jacobsen, S., et al "The Version 1 of the Utah/MIT Dexterous Hand," PROCEEDINGS OF THE 2ND INTERNATIONAL SYMPOSIUM OF ROBOTICS RESEARCH, UJI, Kyoto, Japan, August 1984.
11. Jacobsen, S. et al "Approximately Version 2 of the Utah/MIT Hand," PROC. OF THE 2ND ISRR, Kyoto, Japan, 1984. Published by MIT Press, 1984.
12. McCormick, Malcolm, "A Primer On: Brushless DC Motors," MECHANICAL ENGINEERING, New York, NY, Vol. 110, No. 2, February 1988, pp. 52-57.
13. TORQUE MOTOR ENGINEERING HANDBOOK, Clifton Precision, Litton Systems, Inc., Clifton Heights, PA, 1986, Section 3.
14. HARMONIC DRIVE DESIGNER'S HANDBOOK, Harmonic Drive Division, Emhart Machinery Group, Wakefield, MA, 1986.

ORIGINAL PAGE IS
OF POOR QUALITY

15. "A. D. Little/Sarcos Dexterous Hand Master," product brochure, Arthur D. Little, Inc., Cambridge, MA, 1988
16. "VPL Research DataGlove Model 2 System," product brochure, VPL Research Inc., Redwood City, CA, 1987.
17. Foley, James D., "Interfaces for Advanced Computing," SCIENTIFIC AMERICAN, New York, NY, Vol. 257, No. 4, October 1987, pp. 127-135.
18. Fisher, Scott S., "Telepresence Master Glove Controller for Dexterous Robotic End-Effectors," INTELLIGENT ROBOTS AND COMPUTER VISION - FIFTH IN A SERIES, PROCEEDINGS OF SPIE-THE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING, Vol. 726, Cambridge, Massachusetts, 28-31 October 1986,
19. Abd-Allah, Mahmoud A., "Eigenimage Filtering of Medical Images Using Multispectral Image Sequences," Ph.D. dissertation, University of Toledo, June 1986.
20. Windham, Joe P., M. A. Abd-Allah, et al., "Eigenimage Filtering in MR Imaging," JOURNAL OF COMPUTER ASSISTED TOMOGRAPHY, Vol. 12, No. 1, January/February, 1988, pp. 1-9.
21. Davis, Virgil L., "Computer Hardware and Software for Robotic Control: The Kennedy Space Center (KSC) Robotics Applications Development Laboratory (RADL)," 1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics, May 14, 1987.

CAD-Model-Based Vision for Space Applications

Linda G. Shapiro

Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195
USA

ABSTRACT

A pose acquisition system operating in space must be able to perform well in a variety of different applications including automated guidance and inspections tasks with many different, but known objects. Since the space station is being designed with automation in mind, there will be CAD models of all the objects, including the station itself. The goal of our work is to construct vision models and procedures directly from the CAD models. The system we are designing and implementing must convert CAD models to vision models, predict visible features from a given view point from the vision models, construct view classes representing views of the object, and use the view class model thus derived to rapidly determine the pose of the object from single images and/or stereo pairs.

Keywords: matching, view class, CAD model, relational pyramid

1. Introduction

CAD systems are increasingly becoming an integral component of the manufacturing process in the factories and plants of the United States. Parts can be designed interactively and then automatically machined from the newly created models. As part of the automation process, some manufacturers are buying machine vision systems to inspect the machined parts and to provide the three-dimensional guidance for robots that handle the parts. Currently, most of these systems have to be reprogrammed to recognize each new part; this can involve weeks or months of effort, depending on the task. Since many manufacturers have the CAD models of the parts, the most desirable approach is to produce the vision algorithms directly from the CAD

This research was supported by the National Aeronautics and Space Administration (NASA) through a subcontract from Machine Vision International.

models. This approach has already been recognized as important, and some preliminary work in the area has been done. Examples include the work of Henderson and Bhanu at The University of Utah (Henderson et al., 1986) and that of Ikeuchi at Carnegie-Mellon (Ikeuchi, 1987). One area where vision systems based on CAD models is becoming important is in the United States space program. Since the space station and space vehicles are recent or even current designs, we can expect to have CAD models of these objects to work with. Vision tasks in space such as docking and tracking of vehicles, guided assembly tasks, and inspection of the space station itself for cracks and other problems can rely on model-directed vision techniques.

We are building a computer vision system that uses CAD models and other knowledge about the objects it will view to generate vision models and vision algorithms. The system has two major subsystems: one for offline processing and one for online processing. The offline processing subsystem will handle the task of converting each of the CAD models, which were meant to be used for design and display of objects, to a vision model and a procedure that can be used to recognize, inspect, or manipulate the object. The online processing will use the vision model and procedure to perform the inspection/guidance task.

CAD models range from the standard three-view, orthographic engineering drawings to the more recent constructive solid geometry (CSG) and boundary models produced by solid modeling systems. Since there are algorithms for converting three views to solid models, we will not consider the three view representation, but are working with solid modeling systems. The purpose of most solid modeling systems is to provide an interactive environment in which an engineer can design a part, with procedures by which he or she can display the part in different orientations. A few systems also produce the specifications for the automatic machining of the part. None of these representations is suitable to be directly used by a machine vision system.

A machine vision system needs a model of a part that specifies the features by which it can be recognized from one or more images. The features that will show up and can be extracted from an image are dependent on the

view, the material of the object, and the lighting setup. Machine vision companies that sell software to customers for particular inspection or guidance applications typically have programmers and engineers who spend months developing the best lighting and optics and the algorithms for determining the exact pose of the part and then performing the inspection or guidance task. All of the knowledge and experience of these people is missing in the bare CAD models. Thus the offline system must contain enough knowledge of lighting and materials and the physics of imaging, that it can perform the same task automatically and reliably.

In our overall design, the input to the offline system consists of 1) the CAD model from the geometric modeling system, 2) a knowledge base of information on materials, lighting, and imaging, in a rule-based form, and 3) the specifications for the particular task to be performed. The offline system will transform this information into a vision model and a procedure for that task, to be used by the online system. This paper deals with the vision models being produced and the matching algorithms that will use these models. The topics to be discussed include 1) prediction of images features from object models, 2) representation of the features predicted for a given view and their interrelationships, 3) generation of viewing clusters, and 4) matching an unknown view to the representatives of the view classes derived from the clustering.

2. Predicting Image Features from Object Models

Prediction of image features from object models is important in determining the machine vision algorithms that will detect and extract the features and match to stored models. Ikeuchi (1987) has included prediction in his system that matches range data derived from photometric stereo. Ponce and Chelberg (1987) have worked out the mathematics for predicting the appearance of limbs and cusps of generalized cylinders. We have chosen to work with standard geometric modeling systems of the type used in computer aided design of industrial parts in order to be most compatible with current and potential industrial applications.

There are two major ways of representing three-dimensional objects in today's geometric modeling systems. Constructive solid geometry (CSG) models are built from primitive solids such as spheres, ellipsoids, cylinders, rectangular parallelepipeds, and cones. The primitive solids are combined using operations of union, intersection, and set difference to produce a binary tree structure that represents the object. The idea is conceptually simple and easy for a designer to learn. Furthermore, the objects produced are guaranteed to be physically possible three-dimensional objects. The tree structures can be used to generate wire frame drawings of the object or, with the use of a ray

casting procedure, gray tone and color images of different views with different lightings. Boundary models represent an object by its surfaces and edges. They are more difficult to use for construction of an object, but potentially more flexible, since the surfaces and edges may be represented by B-splines, allowing much more generality. It is possible to construct a boundary representation (BREP) that does not correspond to any physically possible object. Boundary models can also be used to generate wire frame, gray tone, or color images.

We are using three different systems in our present work. The first system, Renaissance, is an experimental CSG modeller being developed by Tony DeRose of the Computer Science Department at the University of Washington. This system accepts CSG input and applies ray casting to produce shaded images. It allows spheres, ellipsoids, cylinders with spherical or ellipsoidal faces, rectangular parallelepipeds, cones, and half planes as primitives. It also allows the user to specify the reflectivity of the surface material and to include any number of point light sources at different positions and with different intensities. Ongoing research on this system is producing the ability to specify areal light sources of various shapes that are more realistic than the point light sources. We are using this system mainly to generate artificial images for display and for image processing.

The second system we are using is PADL-2 (Voelcker and Requicha, 1977) which was developed at Rochester and is now being distributed by Cornell University. PADL-2 is a CSG system, but has the ability to convert the CSG models to BREP. Since vision systems deal with surfaces and edges, this conversion is essential to our work. PADL-2 allows spheres, cylinders, rectangular parallelepipeds, cones, wedges, and tori as primitives, and it does not have as much flexibility as WART in selection of lighting. We are mainly using it for fast wire frame drawings of the objects and for the conversion to BREP.

The third system we are using is CATIA, a joint venture of IBM and Dassault. CATIA can produce a BREP with not only simple surfaces and curves, but also B-spline surfaces and curves. Although we cannot access the CATIA system ourselves, we will be given data from that system via IGES format tapes.

The prediction work consists of extracting data from various geometric modeling systems, storing it in a three-dimensional vision model of our own design, and using the vision model to predict the features that will appear in images. The extraction process is merely a programming task, made difficult or easy by the particular system from which the data must come. The vision model we are currently using is shown in Figure 1. It is a variation of the hierarchical, relational structure we designed for collision avoidance and matching several years ago (Shapiro and Haralick, 1984). The basic data structures employed are the *relational data structure*, which represents entities, and

the *relation*, which represents attributes of and relationships among entities. A relational data structure is a set of named relations. Each relation is a set of N-tuples for some positive integer N. The components of an N-tuple may be atomic or may themselves be relational data structures. Each relational data structure has one distinguished relation called the attribute-value table. The attribute-value table stores the values associated with global attributes of the entity represented by the relational data structure. The other relations often depict the relationships among the primitive pieces of the entity.

The entities shown in Figure 1 are the world, the object, the face, the boundary, the simple arc, the compound arc, the simple surface and the compound surface. The world is made up of objects. The Objects Relation, which is a part of the World Relational Data Structure, is a list of the objects in the world. Each object in the list has a name, a type, a pointer or reference to the relational data structure for that object, and a transformation that can be applied to the points of the object to position it in the world. The attribute-value table of the world contains global information about the world, including, but not limited to the bounding box shown in the figure. An object has a set of faces and two important relations that embody the topology of the object. The Edge/Surface Topology Relation is a list of the three-dimensional (and possibly curved) edges of the object. Each edge in this relation is associated with its two endpoints, the faces to its left and right, the arcs that represent the edge in the boundaries of the two faces, and the angle between the two faces (or an approximation if they are not planar). The Vertex Relation is a list of three-dimensional vertices. Each vertex has associated a name, a location, a list of edges that meet at that vertex, and a transformation.

Conceptually, a face has a surface equation or equations and a set of boundaries that tell what portions of the surface actually belong to the face. These boundaries are listed in the Boundaries Relation, and the surface is referenced as the value of the Surface attribute in the attribute-value table of the face. This is consistent, because the face has only one surface, but it has a set of boundaries. Each boundary is a list of arcs, which can be simple (represented by an equation) or compound (represented by a Compound Arc Relational Data Structure which itself has an Arcs Relation or list of arcs). Similarly, surfaces are either simple surfaces or Compound Surface Relational Data Structures. At all levels of the structure, entities have associated transformations and bounding boxes.

We have used data structures of this general form for model-based vision systems and geographic information systems and have found them to be very flexible and very natural representations of complex structures. The particular variant discussed here stores all the information usually found in any BREP plus the topological information that can help generate constraints for matching in a vision

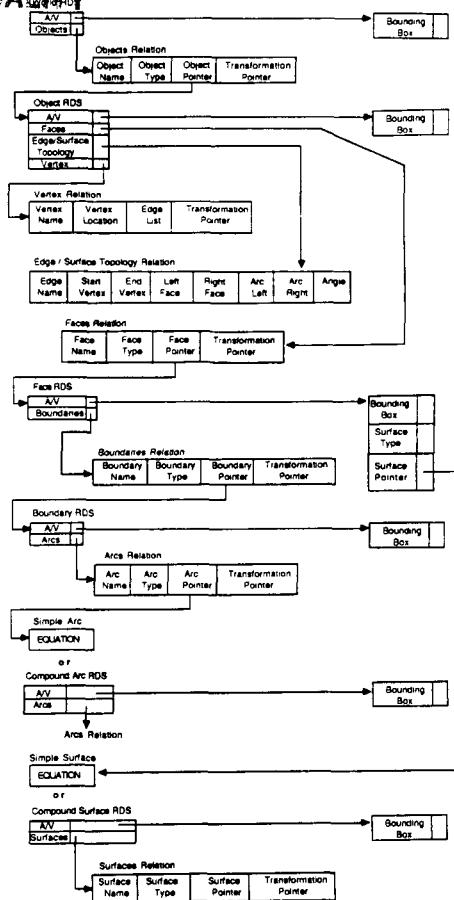


Figure 1 illustrates the relational data structure for a three-dimensional vision model.

system. Later in the work, inspection specifications and other attributes will be added to the models.

Given a relational data structure representing a three-dimensional object, the goal of the prediction module is to predict the features of that object that will appear in an image of the object from a given viewpoint and with a given set of light sources. Visible features can consist of edges between separate surfaces of the object, limbs (loci of points where the line of sight is tangent to the surface), corners, holes, imprinted characters, and anything else we might be able to detect with low- and mid-level vision algorithms. Our initial system will predict edges and limbs that should appear in an image of the object from a particular viewpoint and independent of the lighting. The result will be similar to a wire-frame rendering of the object, and the algorithms used will be variations of standard hidden line/hidden surface algorithms used to produce wire-frames. The main difference will be the maintenance of the relationships between the predicted two-dimensional structures and the three-dimensional entities which produced them. This information will be used later

in the pose estimation procedures.

We are currently beginning the implementation of this part of the system. We have derived the equations of the limbs for the simple CSG primitives (ellipsoid, cylinder, and cone) in closed form. Yet to be derived are the equations and procedures for working with B-spline data. The next step is to investigate and characterize the reasons that an edge or a limb may not show up in an image at all. Not enough contrast between two surfaces due to illumination or even the color and patterns of the surfaces and not enough difference in the orientations of the two surfaces are some of the possible causes.

3. Representation of Features

The prediction system will generate the features that appear in a given view, and those views that produce similar features will be grouped together as one viewing cluster. In order to decide if two views are similar enough to be grouped together, we need a representation for the features and their interrelationships. This representation should be simple enough that the primitive features can be easily accessed and complicated enough that powerful high-level relationships can be represented. This suggests a pyramid structure where simple primitives are represented at the bottom level, and the succeeding levels represent more and more complex relationships among the primitives. Thus the view depicted by this structure can be dealt with at any level of complexity desired. The structure is formally defined as follows.

Let F be a set of detectable primitive features. Each feature $f \in F$ has an associated type T_f and a vector of attributes A_f . A *relational pyramid* of height h over feature set F is a sequence of h relational descriptions $\langle D_0, D_1, \dots, D_{h-1} \rangle$. Description D_0 is a sequence of n_0 relations $\langle R_0^1, \dots, R_0^{n_0} \rangle$, each relation representing one of the primitive types. A pair (f, A_f) belongs to relation R_i^0 if $f \in F$ is a primitive feature of the type represented by R_i^0 and A_f is its vector of attributes. Intuitively, at level 0 of the relational pyramid, each feature is associated with its attributes and is classified as one of several different legal types.

Description D_1 is a sequence of relations $\langle R_1^1, \dots, R_1^{n_1} \rangle$ where each relation R_i^1 represents a relationship among two or more of the level-0 primitives. An attributed tuple of one of these level-1 relations R_i^1 has the form $((N_1, t_1), \dots, (N_n, t_n), A)$ where each N_j is the *name* of a relation $R_{N_j}^0$ at level 0, and the corresponding t_j is a *tuple* of $R_{N_j}^0$. The semantics of $((N_1, t_1), \dots, (N_n, t_n), A) \in R_i^1$ is that the level-0 attributed primitives (t_1, \dots, t_n) which are of types (N_1, \dots, N_n) , are related according to the level-1 relationship R_i^1 , and this level-1 relationship has attribute vector A . This idea can then be extended up the pyramid. At level k , description D_k is a sequence of relations $\langle R_k^1, \dots, R_k^{n_k} \rangle$,

where each relation R_i^k represents a relationship among two or more of the entities from level 0 to level $k-1$. (In the strictest kind of pyramid, they would all be from level $k-1$.) An attributed tuple of such a level- k relation R_i^k has the form $((N_1, t_1), \dots, (N_n, t_n), A)$ where each N_j is the name of a relation $R_{N_j}^{k'}$ at a previous level k' and $t_j \in R_{N_j}^{k'}$ for $j = 1, \dots, n$. The semantics of $((N_1, t_1), \dots, (N_n, t_n), A) \in R_i^k$ is that the attributed primitives (t_1, \dots, t_n) which come from levels 0 to $k-1$ and which are of relational types (N_1, \dots, N_n) are related according to the level- k relationship R_i^k and this level- k relationship has attribute vector A .

Thus the relational pyramid structure allows us to define an object by its attributed primitives, relationships among those primitives, relationships among those relationships, and so on up to some predefined maximal level. It is a hierarchical, relational structure, but the hierarchy is defined on relationships instead of on larger and larger pieces of the object. Having formally defined the structure, we will now show how it can be used to describe a view or a view class of a three-dimensional object.

The level-0 primitives in our current system are straight and curved line segments. Thus in our formalism, $D_0 = \langle \text{straight_segments}, \text{curved_segments} \rangle$. The attribute vector for a straight line segment contains its starting point and its ending point, and the attribute vector for a curved line segment contains its starting point, its ending point, and an interior point which is used in later calculations of relationships.

The level-1 relations represent junctions where two or three segments meet. (This will later be extended to multi-segment junctions.) For junctions where only straight lines meet, we use the standard junction types FORK, ARROW, T, and L. Because we wish to distinguish between the separate lines of each junction, we define a numbering scheme that selects the first line in each junction as the one closest to vertical and then numbers the remaining lines consecutively in clockwise order.

For junctions including at least one curved line, we chose to define a new labeling scheme that helps us to build up relations at the next level of the pyramid. (For an alternate labeling scheme for junction with curves, where junction types represent 3D information rather than just 2D configurations, see Chakravarty (1979).) In our current scheme, a curved segment is considered concave (A) or convex (V) depending on the way it faces the segment previous to it in a clockwise ordering of the segments. (Since our curve segments come from spheres, cylinders, and cones, they will not have inflection points.) The first segment in a junction with a straight line segment and one or two curved line segments is defined to be the straight line segment. The first segment in a junction with two straight line segments and one curved line segment is the straight line segment counterclockwise from the curved segment. If there are no straight lines, the curved segment whose chord joining its start and end points is closest to vertical will be considered

the first segment. The label of a junction then depends on the labels of the two or three segments comprising it, in the ordering in which they are numbered. For example, LA is the label of a junction where a straight line segment connects to a concave curve segment, while LAV is the label of a junction where a straight line segment is followed (in clockwise order) by a concave curve segment which is followed by a convex curve segment.

The relations currently implemented at level 1 of the pyramid represent each of the junction types just described. The LOOP relation, which is also being implemented, consists of sets of segments that together form a minimal closed boundary. Other feasible level-1 relations for view classes would be parallel line segments, colinear line segments, and such spatial relations as above, below, left-of, and right-of (when they are invariant for all views in a view class).

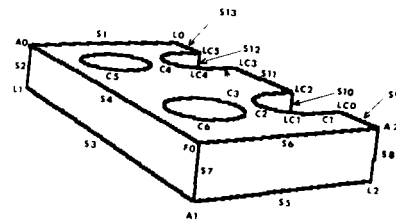
The level-2 relations use level-1 tuples representing attributed junctions and loops and level-0 tuples representing line and curve segments as their primitives. The relations currently being implemented at level 2 are PARALLEL, ANTIPARALLEL, REVERSE, COLINEAR, ADJACENT, and INSIDE. Because these relations are being defined on junctions rather than on line segments, they have special definitions.

The PARALLEL relation consists of attributed sets of parallel junctions. Two straight line junctions $J_1 = (l_1, l_2, \dots, l_n)$ and $J_2 = (l'_1, l'_2, \dots, l'_m)$, $n \leq m$, are *parallel* if there is an order preserving injection $f: J_1 \rightarrow J_2$ satisfying $\text{parallel}(l_i, f(l_i))$, $i = 1, \dots, n$. Two arbitrary junctions $J_1 = (l_1, l_2, \dots, l_n)$ and $J_2 = (l'_1, l'_2, \dots, l'_m)$ are *parallel* if the nonempty subsequences $K_1 \subseteq J_1$ and $K_2 \subseteq J_2$ consisting of only the straight lines are parallel, where the number of straight lines is greater than one.

While two ARROW junctions may be parallel, it is impossible for a FORK junction and an ARROW junction to be parallel, since one line segment of the FORK will point in the opposite direction as the corresponding line segment of the ARROW. Yet this relationship is also important in describing the line drawing as a whole. For this reason, we define the ANTIPARALLEL and REVERSE relations as follows. Two straight line junctions J_1 and J_2 are *antiparallel* if there is a function $f: J_1 \rightarrow J_2$ satisfying $\text{parallel}(l_i, f(l_i))$ or $\text{antiparallel}(l_i, f(l_i))$. Two straight line junctions are *reverse* if they are antiparallel and if $\text{antiparallel}(l_i, f(l_i))$ is true for exactly one pair of corresponding segments. Two junctions are antiparallel (or reverse) if the nonempty subsequences consisting of only straight lines and consisting of at least two straight lines are antiparallel (or reverse). In our implementation, the REVERSE relation corresponds to this definition of reverse, and the ANTIPARALLEL relation corresponds to pairs of junctions that are antiparallel, but not reverse.

The COLINEAR relation consists of attributed sets of colinear junctions. Two junctions are considered colinear if

they are parallel, antiparallel, or reverse and there is one pair of corresponding edges which satisfy the same linear equation. The ADJACENT relation consists of pairs of junctions which are directly connected to each other by a common segment. If above, below, left-of, and right-of are invariant across a view class, these can be used as attributes to the adjacency of the junctions. Finally the INSIDE relation consists of a level-0 primitive and a level-1 loop, where the primitive lies inside the loop. Figure 2 illustrates these concepts for one view class of a three-dimensional object.



LEVEL 2:		
Parallel	Collinear	
{{Fork.F0},{LAL.LC2}}	{{Fork.F0},{Arrow.A0}}	
{{Fork.F0},{LAL.LC5}}	{{Fork.F0},{Arrow.A1}}	
{{Arrow.A2},{LAL.LC2}}	{{Fork.F0},{Arrow.A2}}	
{{Arrow.A2},{LAL.LC5}}	{{L.L1},{Arrow.A0}}	
{{LAL.LC2},{LAL.LC5}}	{{L.L1},{Arrow.A1}}	
	{{L.L2},{Arrow.A1}}	
	{{L.L2},{Arrow.A2}}	
	{{L.L0},{Arrow.A0}}	
	{{L.L0},{Arrow.A2}}	
	{{LAL.LC5},{LAL.LC2}}	
	{{LAL.LC2},{L.L0}}	
	{{LAL.LC5},{Arrow.A2}}	
	{{LAL.LC5},{L.L0}}	
	{{LAL.LC2},{Arrow.A2}}	
Antiparallel	Inside	
{{Fork.F0},{L.L0}}	{{Curve.C5},{Loop.LP1}}	
{{Fork.F0},{L.L1}}	{{Curve.C6},{Loop.LP1}}	
{{Fork.F0},{L.L2}}		
{{Arrow.A0},{Arrow.A1}}		
{{Arrow.A0},{Arrow.A2}}		
{{Arrow.A0},{L.L2}}		
{{Arrow.A1},{Arrow.A2}}		
{{Arrow.A1},{L.L0}}		
{{Arrow.A2},{L.L1}}		
{{L.L1},{LAL.LC2}}		
{{L.L1},{LAL.LC5}}		
Adjacent	Reverse	
{{Fork.F0},{Arrow.A0}}	{{Fork.F0},{Arrow.A1}}	
{{Fork.F0},{Arrow.A1}}	{{Fork.F0},{Arrow.A2}}	
{{Fork.F0},{Arrow.A2}}	{{Fork.F0},{Arrow.A2}}	
{{L.L1},{Arrow.A0}}	{{Arrow.A0},{L.L0}}	
{{L.L1},{Arrow.A1}}	{{Arrow.A0},{L.L1}}	
{{L.L2},{Arrow.A1}}	{{Arrow.A0},{LAL.LC2}}	
{{L.L2},{Arrow.A2}}	{{Arrow.A0},{LAL.LC5}}	
{{L.L0},{Arrow.A0}}	{{Arrow.A1},{L.L1}}	
{{Arrow.A2},{LV.LC0}}	{{Arrow.A1},{L.L1}}	
{{LV.LC3},{LAV.LC4}}	{{Arrow.A1},{LAL.LC2}}	
{{LAV.LC4},{LAL.LC5}}	{{Arrow.A1},{LAL.LC5}}	
{{LAL.LC5},{L.L0}}	{{Arrow.A2},{L.L0}}	
{{LV.LC0},{LAV.LC1}}	{{Arrow.A2},{L.L2}}	
{{LAV.LC1},{LAL.LC2}}		
{{LAL.LC2},{LV.LC3}}		
LEVEL 1:		
Loops		
LP0 {S4.S1.S13.C4.C3.S11.C2.C1.S9.S6}		
LP1 {S4.S2.S3.S7}		
LP2 {S6.S8.S5.S7}		
LP3 C2.S10		
LP4 {C4.S12}		
LP5 {C5}		
LP6 {C6}		
LV_Junctions	LAV_Junctions	LAL_Junctions
LC0 {S9.C1}	LC1 {S10.C1.C2}	LC2 {S10.S11.C2}
LC3 {S11.C3}	LC4 {S12.C3.C4}	LC5 {S13.S12.C4}
L_Junctions	Fork_Junctions	Arrow_Junctions
L0 {S1.S13}	F0 {S4.S6.S7}	A0 {S1.S2.S4}
L1 {S2.S3}		A1 {S3.S7.S5}
L2 {S8.S5}		A2 {S9.S8.S6}
LEVEL 0:		
Straight_Segments {S1.S2.S3.S4.S5.S6.S7.S8.S9.S10.S11.S12.S13}		
Curved_Segments {C1.C2.C3.C4.C5.C6}		

Figure 2b illustrates the pyramid structure for the view class shown in Figure 2a with attribute information suppressed for simplicity.

suggests that two views belong in the same view class if their relational distance (Shapiro and Haralick, 1985) is similar enough. Thus finding view classes consists of finding relational distances between pairs of pyramid structures representing view points selected from a tessellated Gaussian sphere and clustering them, based on these relational distances. This is all part of the offline processing of the object.

5. Matching Unknown Views to View Class Descriptions

When an image is taken of a known 3D object from an unknown view, the first step before inspection or guidance is to determine the pose (position and orientation) of the object. To achieve this with a view class model, the vision system must first determine the correct view class, find the correspondences between the features extracted from the image and those in the view class representation, use the

links between 3D features and view class features to find the correspondence between extracted features and 3D features, and use this correspondence to find the pose of the object.

It is desirable to determine the correct view class as rapidly as possible. Chakravarty and Freeman (1982) represented a view class by a vector containing the number of junctions of each junction type. They used the values in the vector to select the best view class, relying especially on the most frequent junction type. We feel that this approach, while simple and rapid, will not work very well when some of the segments do not show up in the image, causing missing or erroneous junction types, or when extra segments appear in the image. Ikeuchi (1987), using range data, created an interpretation tree during his offline processing phase. The interpretation tree was a decision tree used to select the best view class, depending on the values of various measurements. We will consider this approach in the future, but initially, we are trying a simple idea based on the Hough transform, which we consider promising.

Suppose each view class is represented by a relational pyramid structure. For each relational pyramid, we can easily derive a *summary pyramid* structure. Where the relational pyramid has a relation R with c tuples $\{(N_1, t_{1,j}), (N_2, t_{2,j}), \dots, (N_n, t_{n,j}), A \mid j = 1, \dots, c\}$, the summary pyramid has a corresponding relation R with a single tuple $((N_1, N_2, \dots, N_n), c)$ representing those c tuples. For example, if the parallel relation has 4 tuples of the form $((FORK, f), (ARROW, a))$, then the parallel summary relation has one tuple $((FORK, ARROW), 4)$. This is done for each relation and at each level of the pyramid. At level 0, the summary is just the count c of how many primitive features there are of each type. Figure 3 illustrates the summary structure for the relational pyramid of Figure 2b. Note that we have simplified the level-1 summary structures for readability. The names of these level-1 relations actually indicate the types of the primitives.

Along with the original relational pyramid structures and the summary structures, the online system requires one more structure to be produced by the offline system: the *index structure*. The index allows direct access, given a summary tuple of the form $((N_1, N_2, \dots, N_n), c)$, to a list of all view classes that have this tuple in their summary structures. It keeps an evidence accumulator for each view class, initialized to zero. For exact matching, the online system would traverse the summary structure derived from the unknown view, and for each tuple in the summary structure, it would add one to the accumulators of all the view classes on the list attached to that tuple in the index. The view class or classes with maximal evidence would be selected.

Exact matching will produce erroneous results, due to missing and extra segments. Our current solution is to actually model the "erroneous" views associated with a view class, producing a separate summary for each. We also associate a probability with each perfect and each erroneous

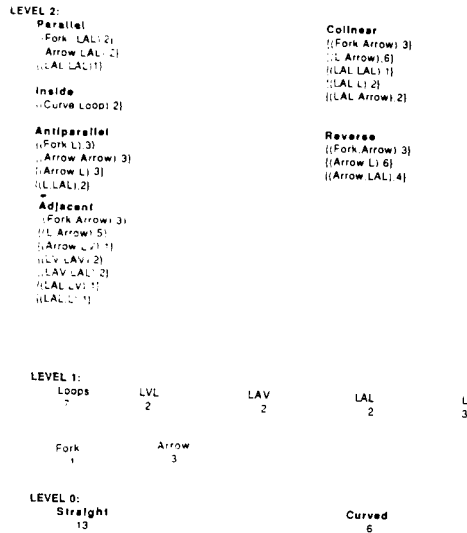


Figure 3 illustrates a summary structure for the relational pyramid of Figure 2b.

view; the probabilities are now chosen by the experimenter, but will eventually come from the prediction module. Using the summary matching described above with the enlarged set of summaries, we can select one or more view classes. A Bayesian analysis will then tell us which is the most probable view.

Once a view class has been selected, we must determine the correspondence between the primitives of its relational pyramid and those of the unknown view's relational pyramid. Since the relational pyramid structure is a highly constrained, relational representation of a view class or view, we expect that a backtracking tree search, using discrete relaxation, will be able to rapidly find the best mapping from view class structure to view structure. In particular, the higher-level relations are expected to aid in efficient pruning of the tree. Furthermore, it is not necessary to find a correspondence for every primitive feature of a view class. We need only find enough correspondences to reliably compute the pose of the object.

6. Summary

We have described a system for pose acquisition. The system consists of modules for predicting features that will appear in 2D images from 3D models, clustering views into view classes, and using the view classes to rapidly determine the pose of the object. The system is currently being implemented in the Intelligent Systems Laboratory at the University of Washington.

References

1. Chakravarty, I., "A Generalized Line and Junction Labelling Scheme with Application to Scene Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 2, 1979, pp. 202-205.
2. Chakravarty, I. and H. Freeman, "Characteristic Views as a Basis for Three-Dimensional Object Recognition", *Proceedings of SPIE 336 (Robot Vision)*, 1982, pp. 37-45.
3. Henderson, T., C. Hansen, A. Samal, C.C. Ho, and B. Bhanu, "CAGD-Based 3-D Visual Recognition", *Proceedings of the Eighth International Conference on Pattern Recognition*, Paris, October 1986, pp. 230-232.
4. Ikeuchi, K. "Generating an Interpretation Tree From a CAD Model for 3D-Object Recognition in Bin-Picking Tasks", *International Journal of Computer Vision*, 1987, pp. 145-165.
5. Korn, M. and C. Dyer, "3-D Multiview Object Representations for Model-Based Object Recognition", *Pattern Recognition*, Vol. 20, No. 1, 1987, pp. 91-103.
6. Ponce, J. and D. Chelberg, "Finding the Limbs and Cusps of Generalized Cylinders", *International Journal of Computer Vision*, Vol. 1, No. 3, 1987, pp. 195-210.
7. Shapiro, L.G. and R. M. Haralick, "A Metric for Comparing Relational Descriptions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 1, 1985, pp. 90-94.
8. Shapiro, L.G. and R. M. Haralick, "A Hierarchical Relational Model for Automated Inspection Tasks", *First IEEE Conference on Artificial Intelligence Applications*, Denver, Dec. 1984, pp. 207-210.
9. Voelker, H.B. and A. A. Requicha, "Geometric Modeling of Mechanical Parts and Processes", *Computer*, Vol. 10, No. 12, Dec. 1977, pp. 48-57.

DECISION TREE ANALYSIS OF SPACECRAFT MOTION

Stanley Sternberg
Richard Toptani
Chanchad Chatterjee
Machine Vision International

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

MODEL BASED PERSPECTIVE PROJECTION EXPERT

Robert M. Haralick
University of Washington

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

SENSOR FUSION OF RANGE AND REFLECTANCE DATA FOR OUTDOOR SCENE ANALYSIS *

In So Kweon

Martial Hebert

Takeo Kanade

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In recognizing objects in an outdoor scene, range and reflectance (or color) data provide complementary information. This paper presents the results of experiments in recognizing outdoor scenes containing roads, trees, and cars from the CMU Navlab (Navigation Laboratory) project. The recognition program uses range and reflectance data obtained by a scanning laser range finder, as well as color data from a color TV camera. After segmentation of each image into primitive regions, models of objects are matched using various properties.

1 Introduction

In order to be able to handle a variety of environments and tasks, a mobile robot must be able to extract features from its visual sensors that enable it to correctly interpret its environment. Many sensing strategies are now possible, from standard video cameras to sophisticated ranging systems. A single sensing modality cannot provide enough information to interpret outdoor scenes however. Geometrical data from ranging systems is necessary for describing the shapes of the observed objects, but some type of reflectance data is also necessary to properly analyze their physical properties, such as terrain type or surface markings. Combining different types of sensor data is not an easy task however. It involves the combination of data sets that are measured by sensors with different characteristics of field of view, range, and accuracy. It also involves the combination of sets of informations of different nature that have been extracted using very different algorithms, such as the combination of surface patches and color edges, for example.

In this paper, we investigate ways to combine geometrical informations from a laser range finder with physical informations from a color camera and an active reflectance sensor (Actually

the reflectance images are also provided by the range sensor). We demonstrate the approaches to the combination of those sensors in two examples. The first one concerns the analysis of outdoor for the recognition of natural objects, such as trees, for which only weak models exist. The second one is the recognition of landmarks for which an accurate geometric model is available. In both cases, the combination of shape and reflectance information provides a better, more reliable, interpretation of the sensor data. We have implemented all the techniques described in this paper on the CMU Navlab (Navigation Laboratory) which is a self-contained mobile robot designed for navigation in outdoor terrain [12].

2 Description of the sensors

In this Section, we describe the geometry and the outputs of the two sensors that we use: a laser range finder, and a color camera. Even though some of the characteristics are fairly specific to the particular sensor, the geometries and noise models of the sensors are representative of a wide range of existing visual sensors.

2.1 The range and active reflectance sensors

The basic principle of active sensing techniques is to observe the reflection of a reference signal (sonar, laser, radar...etc.) produced by an object in the environment in order to compute the distance between the sensor and that object. In addition to the distance, the sensor may report the intensity of the reflected signal which is related to physical surface properties of the object. In accordance with tradition, we will refer to this type of intensity data as "reflectance" data even though the quantity measured is not the actual reflectance coefficient of the surface.

Active sensors are attractive to mobile robots researchers for two main reasons: first, they provide range data without the computation overhead associated with conventional passive techniques such as stereo vision, which is important in time critical applications such as obstacle detection. Second, it is largely insensitive to outside illumination conditions, simplifying considerably the image analysis problem. This is especially important for images of outdoor scenes in which illumination cannot be controlled or predicted. For example, the active reflectance images of outside scenes do not contain any shadows from the sun. In

*This research was sponsored in part by the Defense Advanced Research Projects Agency, DoD, through ARPA Order 5351, monitored by the US Army Engineer Topographic Laboratories under contract DACA76-85-C-0003, by the National Science Foundation contract DCR-8604199, by the Digital Equipment Corporation External Research Program, and by NASA grant NAGW-1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

addition, active range finding technology has developed to the extent [1] that makes it realistic to consider it as part of practical mobile robot implementations in the short term.

The range sensor is a time-of-flight laser range finder developed by the Environmental Research Institute of Michigan (ERIM). The basic principle of the sensor is to measure the difference of phase between a laser beam and its reflection from the scene [7]. A two-mirror scanning system allows the beam to be directed anywhere within a $30^\circ \times 80^\circ$ field of view. The data produced by the ERIM sensor is a 64×256 range image, the range is coded on eight bits from zero to 64 feet, which corresponds to a range resolution of three inches. In addition to range images, the sensor also produces active reflectance images of the same format ($64 \times 256 \times 8$ bits), the reflectance at each pixel encodes the energy of the reflected laser beam at each point. Figure 1 shows a pair of range and reflectance images of an outdoor scene.

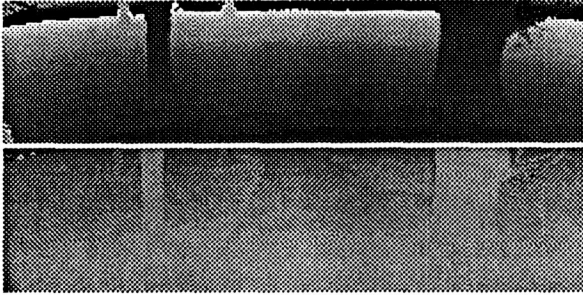


Figure 1: Range and reflectance images

The position of a point in a given coordinate system can be derived from the measured range and the direction of the beam at that point. We usually use the Cartesian coordinate system shown in Figure 2, in which case the coordinates of a point measured by the range sensor are given by the equations:

$$x = D \sin \phi \cos \theta \quad (1)$$

$$y = D \cos \phi \cos \theta \quad (2)$$

$$z = D \sin \theta \quad (3)$$

where ϕ and θ are the vertical and horizontal angular angles of the beam direction. The two angles are derived from the row and column position in the range image (r, c), by the equations:

$$\theta = \theta_0 + c \times \Delta\theta$$

$$\phi = \phi_0 + r \times \Delta\phi \quad (4)$$

where θ_0 (resp. ϕ_0) is the starting horizontal (resp. vertical) scanning angles, and $\Delta\theta$ (resp. $\Delta\phi$) is the angular step between to consecutive columns (resp. rows). Figure 3 shows an overhead view of the scene of Figure 1, the coordinates of the points are computed using Equ. (4).

As is the case with any sensor, the range sensor returns values that are measured with a limited resolution which are corrupted by measurement noise. In the case of the ERIM sensor, the main source of noise is due to the fact that the laser beam is not a

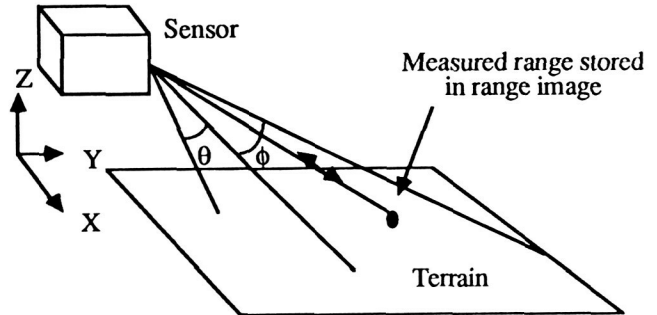


Figure 2: Geometry of the range sensor

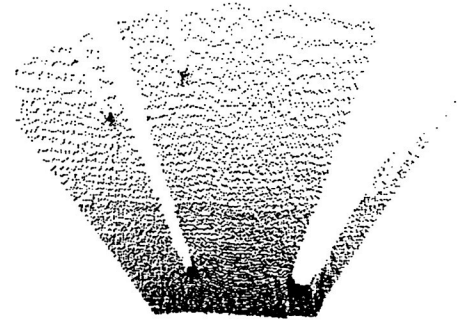


Figure 3: Overhead view

line in space but rather a cone whose opening is a 0.5° solid angle (the instantaneous field of view). The value returned at each pixel is actually the average of the range of values over a 2-D area, the *footprint*, which is the intersection of the cone with the target surface. Simple geometry shows that the area of the footprint is proportional to the square of the range at its center. As a result, the accuracy of the sensor degrades rapidly as the measured points are further away from the sensor which makes the feature extraction a difficult task. The footprint affects all pixels in the image.

There are other effects that produce distortions only at specific locations in the image. The main effect is known as the "mixed point" problem in which the laser footprint crosses the edge between two objects that are far from each other. In that case, the returned range value is some combination of the range of the two objects but does not have any physical meaning. This problem makes the accurate detection of occluding edges more difficult. Another effect is due to the reflectance properties of the observed surface; if the surface is highly specular then no laser reflection can be observed. In that case the ERIM sensor returns a value of 255. This effect is most noticeable on man-made objects that contain a lot of polished metallic surfaces.

2.2 The video camera

The video camera is a standard color vidicon camera equipped with wide-angle lenses. The color images are 480 rows by 512 columns, each band is coded on eight bits. The wide-angle lens induces a significant geometric distortion, that is, the relation between a point in space and its projection on the image plane does not obey the laws of the standard perspective transformation. We alleviate this problem by first transforming the actual image into an "ideal" image: if (R, C) is the position in the real image, then the position (r, c) in the ideal image is given by:

$$r = f_r(R, C), c = f_c(R, C) \quad (5)$$

where f_r and f_c are third order polynomials. This correction is cheap since the right-hand side of (5) can be put in lookup tables. The actual computation of the polynomial is described in [9]. The geometry of the ideal image obeys the laws of the perspective projection in that if $P = [x, y, z]^T$ is a point in space, and (r, c) is its projection in the ideal image plane, then:

$$r = fx/z, c = fy/z \quad (6)$$

where f is the focal length. In the rest of the paper, row and column positions will always refer to the positions in the ideal image, so that perspective geometry is always assumed.

3 Merging range and video images

In order to merge range and color data, we have to store pixels from the two sensors into a common representation, the "*colored-range*" image [10]. Each pixel of a colored-range image contains a color value (*red, green, blue*) from the video camera as well as the position (x, y, z) of the measured point in space as derived from the geometry of the range sensor. By "image", we do not

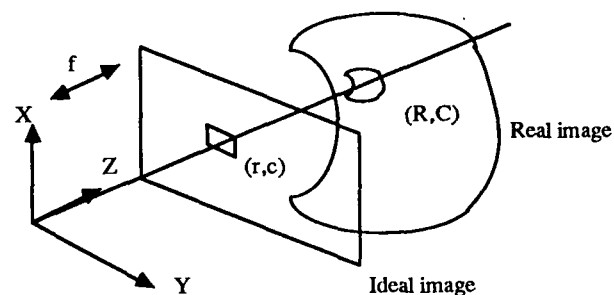


Figure 4: Geometry of the video camera

necessarily mean an array that stores those 6-dimensional pixels, but rather a set of functions that provide access to the range and color data at any point as described in Section 3.1.4. Our first task for building a colored-range image is to express the points in video and range image in a common reference frame, that is to solve the registration problem.

3.1 The registration problem

Range sensor and video cameras have different fields of view, orientations, and positions. In order to be able to merge data from both sensors, we first have to estimate their relative positions, this is known as the calibration, or registration problem (Figure 5). We approach the problem as a minimization problem in which pairs of pixels are selected in the range and video images. The pairs are selected so that each pair is the image of a single point in space as viewed from the two sensors. The problem is then to find the best calibration parameters given these pairs of points. The problem is further divided into two steps: we first use a simple linear least-squares approach to find a rough initial estimate of the parameters, and then apply a non-linear minimization algorithm to compute an optimal estimate of the parameters.

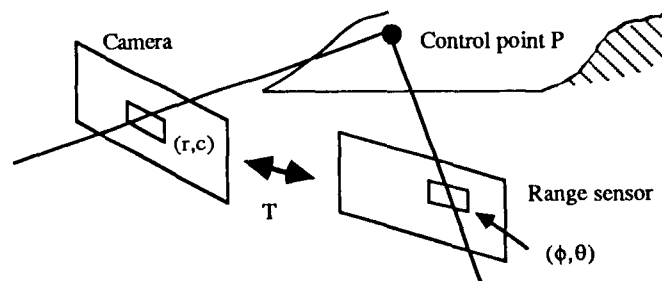


Figure 5: Geometry of the calibration problem

3.1.1 The calibration problem as a minimization problem

Let P_i be a point in space, with coordinates P_i^r with respect to the range sensor, and coordinates P_i^v with respect to the video camera. The relationship between the two coordinates is:

$$P_i^r = RP_i^v - T \quad (7)$$

where R is a rotation matrix, and T is a translation vector. R is a non-linear function of the orientation angles of the camera: pan (α), tilt (β), and rotation (γ). P_i^r can be computed from a pixel location in the range image. P_i^c is not completely known, it is related to the pixel position in the video image by the perspective transformation:

$$z_i^c r_i = f x_i^c \quad (8)$$

$$z_i^c c_i = f y_i^c \quad (9)$$

where f is the focal length. Substituting (7) into (8) and (9) we get:

$$R_x P_i^r r_i - T_x r_i - f R_x P_i^c + T_x' = 0 \quad (10)$$

$$R_y P_i^r c_i - T_y c_i - f R_y P_i^c + T_y' = 0 \quad (11)$$

where R_x , R_y , and R_z are the row vectors of the rotation matrix R , and $T_x' = fT_x$, $T_y' = fT_y$.

We are now ready to reduce the calibration problem to a least-squares minimization problem. Given n points P_i , we want to find the transformation (R, T) that minimizes the left-hand sides of equations (10) and (11). We first estimate T by a linear least-squares algorithm, and then compute the optimal estimate of all the parameters.

Assuming that we have an estimate of the orientation R , we first want to estimate the corresponding T . The initial value of R can be obtained by physical measurements using inclinometers. Under these conditions, the criterion to be minimized is:

$$\sum_{i=1}^n (A_i - T_x B_i - f C_i + T_x')^2 + (D_i - T_y E_i - f F_i + T_y')^2 \quad (12)$$

where $A_i = R_x P_i^r r_i$, $B_i = r_i$, $C_i = R_x P_i^c$, $D_i = R_y P_i^r c_i$, $E_i = c_i$, and $F_i = R_y P_i^c$ are known and T_x , T_y , T_x' , T_y' , f are the unknowns.

Equation (12) can be put in matrix form:

$$C = \|U - AV\|^2 + \|W - BV\|^2 \quad (13)$$

where $V = [T_x', T_y', T_x, T_y, f]^T$ is the vector of unknowns, and A, U, W, B are matrices that are functions of the known quantities only. The minimum for the criterion of Equation (13) is attained at the parameter vector:

$$V = (A^T A + B^T B)^{-1} (A^T U + B^T W) \quad (14)$$

Once we have computed the initial estimate of V , we have to compute a more accurate estimate of (R, T) . Since R is a function of (α, β, γ) , we can transform the criterion from equation (12) into the form:

$$C = \sum_{i=1}^n \|I_i - H_i(S)\|^2 \quad (15)$$

where I_i is the 2-vector representing the pixel position in the video image, $I_i = [r_i, c_i]^T$, and S is the full vector of parameters, $S = [T_x', T_y', T_x, T_y, f, \alpha, \beta, \gamma]^T$. We cannot directly compute C_{min} since the functions H_i are non-linear, instead we linearize C by using the first order approximation of H_i [8] thus reducing the problem to a linear least-squares minimization that can be solved directly. The procedure is iterated until S cannot be improved any further.

3.1.2 Implementation and performance

The implementation of the calibration procedure follows the steps described above. Pairs of corresponding points are selected in a sequence of video and range images. We typically use twenty pairs of points carefully selected at interesting locations in the image (e.g. corners). An initial estimate of the camera orientation is $(0, \beta, 0)$, where β is physically measured using an inclinometer. The final estimate of S is usually obtained after less than ten iterations. This calibration procedure has to be applied only once, as long as the sensors are not displaced.

Once we have computed the calibration parameters, we can merge range and video images into a colored-range image. Instead of having one single fusion program, we implemented this as a library of fusion functions that can be divided in two categories:

1. Range \rightarrow video: This set of functions takes a pixel or a set of pixels (r^c, c^c) in the range image and computes the location (r^r, c^r) in the video image. This is implemented by directly applying Equations (10) and (11).
2. Video \rightarrow range: This set of functions takes a pixel or a set of pixels (r^r, c^r) in the video image and computes the location (r^c, c^c) in the range image. The computed location can be used in turn to compute the location of a intensity pixel in 3-D space by directly applying Equation (4). The algorithm for this second set of functions is more involved because a pixel in the video image corresponds to a line in space (Figure 4) so that Equations (10) and (11) cannot be applied directly. More precisely, a pixel (r^r, c^r) corresponds, after transformation by (R, T) , to a curve C in the range image. C intersects the image at locations (r^c, c^c) , the algorithm reports the location (r^c, c^c) that is the minimum among all the range image pixels that lie on C of the distance between (r^c, c^c) and the projection of (r^r, c^r) in the video image (using the first set of functions). The algorithm is summarized on Figure 6.

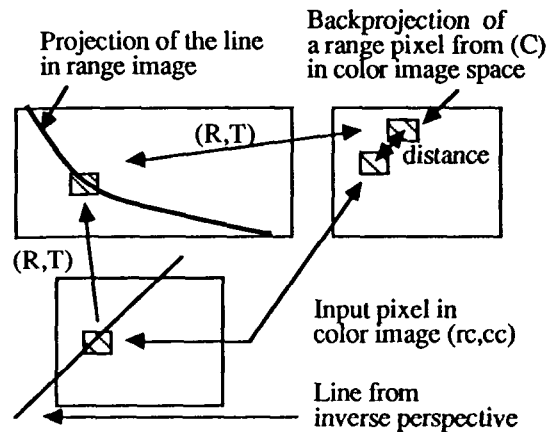


Figure 6: Geometry of the "video \rightarrow range" transformation

Figure 7 shows the colored-range image of a scene of stairs and sidewalks, the image is obtained by mapping the intensity

values from the color image onto the range image. Figure 8 shows a perspective view of the colored-range image. In this example [10], we first compute the location of each range pixel (r^x, c^x) in the video image, and then assign the color value to the 64×256 colored-range image. The final display is obtained by rotating the range pixels, the coordinates of which are computed using Equation (4).

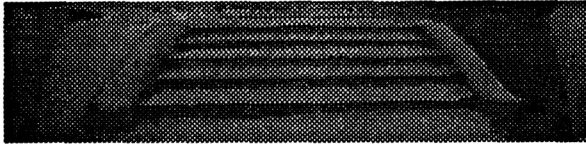


Figure 7: Colored-range image of stairs

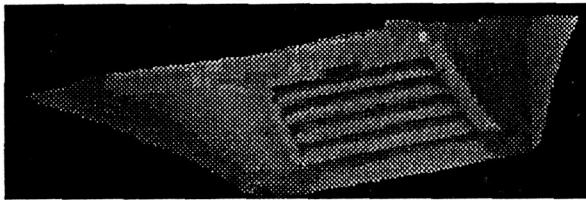


Figure 8: Perspective view of registered range and color images

3.2 Application to outdoor scene analysis

An example of the use of the fusion of range and video images is outdoor scene analysis [6,10] in which we want to identify the main components of an outdoor scene, such as trees, roads, grass, etc. The colored-range image concept makes the scene analysis problem easier by providing data pertinent to both geometric information (e.g. the shape of the trees) and physical information (e.g. the color of the road).

3.2.1 Feature extraction from a colored-range image

The features that we extract from a colored-range image must be related to two types of information: the shapes and the physical properties of the observed surfaces.

The geometric features are used to describe the shape of the objects in the scene. We propose to use two types of features: regions that correspond to smooth patches of surface, and edges that correspond either to transitions between regions, or to transitions between objects (occluding edges). Furthermore, we must be able to describe the features in a compact way. One common approach is to describe the regions as quadric patches, and the edges as sets of tri-dimensional line segments. More sophisticated descriptions are possible [4], such as bicubic patches or curvature descriptors. We use simpler descriptors since the range data is relatively low resolution, and we do not have the type of accurate geometric model that is suited for using higher order geometric descriptors. The descriptors attached to each geometric feature are:

- The parameters describing the shape of the surface patches. That is the parameters of the quadric surface that approximate each surface patch.
- The shape parameters of the surface patches such as center, area, and elongations.
- The 3-D polygonal description of the edges.
- The 3-D edge types: convex, concave, or occluding.

The surface patches are extracted by fitting a quadric of equation $X'AX + B'X + C = 0$ to the observed surfaces, where X is the Cartesian coordinate vector computed from a pixel in the range image. The fitting error,

$$E(A, B, C) = \sum_{X_i \in \text{patch}} [X_i'AX_i + B'X_i + C]^2 \quad (16)$$

is used to control the growing of regions over the observed surfaces. The parameters A, B, C are computed by minimizing $E(A, B, C)$ as in [3].

The features related to physical properties are regions of homogeneous color in the video image, that is regions within which the color values vary smoothly. The choice of these features is motivated by the fact that an homogeneous region is presumably part of a single scene component, although the converse is not true as in the case of the shadows cast by an object on an homogeneous patch on the ground. The color homogeneity criterion we use is the distance $(X - m)' \Sigma^{-1} (X - m)$ where m is the average mean value on the region, Σ is the covariance matrix of the color distribution over the region, and X is the color value of the current pixel in (red, green, blue) space. This is a standard approach to color image segmentation and pattern recognition. The descriptive parameters that are retained for each region are:

- The color statistics (m, Σ).
- The polygonal representation of the region border.
- Shape parameters such as center or moments.

The range and color features may overlap or disagree. For example, the shadow cast by an object on a flat patch of ground would divide one surface patch into two color regions. It is therefore necessary to have a cross-referencing mechanism between the two groups of features. This mechanism provides a two-way direct access to the geometric features that intersect color features. Extracting the relations between geometric and physical features is straightforward since all the features are registered in the colored-range image.

An additional piece of knowledge that is important for scene interpretation is the spatial relationships between features. For example, the fact that a vertical object is connected to a large flat plane through a concave edge may add evidence to the hypothesis that this object is a tree. As in this example, we use three types of relational data:

- The list of features connected to each geometric or color feature.
- The type of connection between two features (convex/concave/occluding) extracted from the range data.
- The length and strength of the connection. This last item is added to avoid situations in which two very close regions become accidentally connected along a small edge.

3.2.2 Scene interpretation from the colored-range image

Interpreting a scene requires the recognition of the main components of the scene such as trees or roads. Since we are dealing with natural scenes, we cannot use the type of geometric matching that is used in the context of industrial parts recognition [4]. For example, we cannot assume that a given object has specific quadric parameters. Instead, we have to rely on "fuzzier" evidence such as the verticality of some objects or the flatness of others. We therefore implemented the object models as sets of properties that translate into constraints on the surfaces, edges, and regions found in the image. For example, the description encodes four such properties:

- *P1*: The color of the trunk lies within a specific range \Rightarrow constraint on the statistics (m, Σ) of a color region.
- *P2*: The shape of the trunk is roughly cylindrical \Rightarrow constraint on the distribution of the principal values of the matrix A of the quadric approximation.
- *P3*: The trunk is connected to a flat region by a concave edge \Rightarrow constraint on the neighbors of the surface, and the type of the connecting edge.
- *P4*: The tree has two parallel vertical occluding edges \Rightarrow constraint on the 3-D edges description.

Other objects such as roads or grass areas have similar descriptions. The properties P_{ij} of the known object models M_j are evaluated on all the features F_k extracted from the colored-range image. The result of the evaluation is a score S_{ijk} for each pair (P_{ij}, F_k) . We cannot rely on individual scores since some may not be satisfied because of other objects, or because of segmentation problems. In the tree trunk example, one of the lateral occluding edges may itself be occluded by some other object, in which case the score for *P4* would be low while the score for the other properties would still be high. In order to circumvent this problem, we first sort the possible interpretations M_j for a given feature F_k according to all the scores $(S_{ij})_i$. In doing this, we ensure that all the properties contribute to the final interpretation and that no interpretations are discarded at this stage while identifying the most plausible interpretations.

We have so far extracted plausible interpretations only for individual scene features F_k . The final stage in the scene interpretation is to find the interpretations (M_{jk}, F_k) that are globally consistent. For example, property *P3* for the tree implies a constraint on a neighboring region, namely that this has to be a flat ground region. Formally, a set of consistency constraints C_{mn} is associated with each pair of objects (M_m, M_n) . The C_{mn} constraints are propagated through the individual interpretations (M_{jk}, F_k) by using the connectivity information stored in the colored-range feature description. The propagation is simple considering the small number of features remaining at this stage.

The final result is a consistent set of interpretations of the scene features, and a grouping of the features into sets that correspond to the same object. The last result is a by-product of the consistency check and the use of connectivity data. Figure 9 shows the color and range images of a scene which contains a road, a couple of trees, and a garbage can. Figure 10 shows a display of the corresponding colored-range image in which the white pixels

are the points in the range image that have been mapped into the video image. This set of points is actually sparse because of the difference in resolutions between the two sensors, and some interpolation was performed to produce the dense regions of Figure 10.

Only a portion of the image is registered due to the difference in field of view between the two sensors (60° for the camera versus 30° in the vertical direction for the range sensor). Figure 12 shows a portion of the image in which the edge points from the range image are projected on the color image. The edges are interpreted as the side edges of the tree and the connection between the ground and the tree. Figure 11 shows the final scene interpretation. The white dots are the main edges found in the range image. The power of the colored-range image approach is demonstrated by the way the road is extracted. The road in this image is separated into many pieces by strong shadows. Even though the shadows do not satisfy the color constraint on road region, they do perform well on the shape criterion (flatness), and on the consistency criteria (both with the other road regions, and with the trees). The shadows are therefore interpreted as road regions and merge with the other regions into one road region. This type of reasoning is in general difficult to apply when only video data is used unless one uses stronger models of the objects such as an explicit model of a shadowed road region. Using the colored-range image also makes the consistency propagation a much easier task than in purely color-based scene interpretation programs [11].

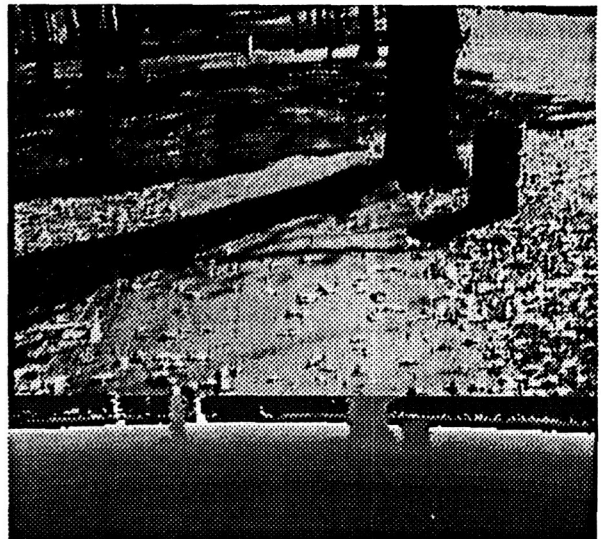


Figure 9: Color and range images of an outdoor scene

4 Merging range and active reflectance images

In the previous section we discussed the fusion of data from a video camera and a range sensor. We now discuss the fusion

ORIGINAL PAGE IS
OF POOR QUALITY

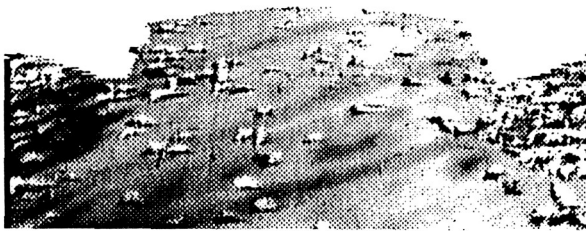


Figure 10: A view of the corresponding colored-range image



Figure 11: Final scene interpretation



Figure 12: Edge features from the colored-range image

of two types of data from the ERIM sensor: range and active reflectance. This problem is somewhat simpler since the two images are already registered by the sensor itself. We will therefore focus our attention on the analysis of the active reflectance images, and the application of simultaneous range and reflectance processing to object recognition.

4.1 Correction of active reflectance images

A reflectance image from the ERIM sensor is an image of the energy reflected by the reflection of a laser beam. Unlike conventional intensity images, this data provide us information which is to a large extent independent of the environmental illumination. In particular, the reflectance images contain no shadows from outside illumination. The measured energy depends also on the shape of the surface, and its distance to the sensor. We correct the image so that the pixel values are functions of the material reflectance only. The measured energy, P_{return} , depends on the specific material reflectance, ρ , the range, D , and the angle of incidence, γ :

$$P_{return} = \frac{K\rho \cos \gamma}{D^2} \quad (17)$$

Due to the wide range of P_{return} , the value actually reported in the reflectance image is compressed by using a log transform. That is, the digitized value, P_{image} is of the form [14]:

$$P_{image} = A \log(\rho \cos \gamma) + B \log D \quad (18)$$

where A and B are constants that depend only on the characteristics of the laser, the circuitry used for the digitization, and the physical properties of the ambient atmosphere. Since A and B cannot be computed directly, we use a calibration procedure in which a homogeneous flat region is selected in a training image, we then use the pixels in this region to estimate A and B by least-squares fitting Equ. (18) to the actual reflectance/range data. Given A and B , we correct subsequent images by:

$$P_{new-image} = (P_{image} - B \log D)/A \quad (19)$$

The value $P_{new-image}$ depends only on the material reflectance and the angle of incidence. This a sufficient approximation for our purposes since for smooth surfaces, such as smooth terrain, the

$\cos \gamma$ factor does not vary widely. For efficiency purposes, the right-hand side of 19 is precomputed for all possible combinations (P_{image}, D) and stored in a lookup table. Figure 1 shows an example of ERIM image and Figure 13 shows the resulting corrected image.



Figure 13: Corrected reflectance image

4.2 Application to 3-D feature extraction for object recognition

We now tackle the problem of fusing range and reflectance data for recognizing objects for landmark-based robot navigation [5]. The problem is different from the previous scene description problem in several respects. First of all, we assume that we have a geometric model of the landmark. Furthermore, we want to not only identify the object in the scene, but also to compute its position and attitude. It is critical to extract accurate geometric features from the images in order to relate the observed scene to the stored models. The fusion of range and reflectance data is used to improve the quality of the surface description extracted from the image data.

The 3-D features that are needed for object recognition are connected surface patches. Each patch corresponds to a smooth portion of the surface and is approximated by a parameterized surface. In addition to the parameters and the neighbors, each region has two uncertainty factors: σ_a , and σ_d . σ_a is the variance of the angle between the measured surface normal and the surface normal of the approximating surface at each point. σ_d is the variance of the distance between the measured points and the approximating surface. Those two attributes are used in the object recognition algorithm.

Several range image segmentation techniques have been proposed in previous works [4]. These techniques are based either on clustering in some parameter space, or region growing using the smoothness of the surface. We chose to combine both approaches into a single segmentation algorithm. The algorithm first attempts to find groups of points that belong to the same surface, and then uses these groups as seeds for region growing, so that each group is expanded into a smooth connected surface patch. The smoothness of a patch is evaluated by fitting a surface, plane or quadric, in the least-squares sense.

The strategy for expanding a region is to merge the best point at the boundary of each region at each step. This strategy guarantees a near optimal segmentation. It has, however, two major drawbacks: it may be computationally expensive, and it may lead to errors due to sensor errors on isolated points, such as mixed points. To alleviate these problems, we use a multi-resolution approach. We first apply the segmentation to a reduced image in which each pixel corresponds to a $n \times n$ window in the original

image, n being the reduction factor. This first, low-resolution, step produces a conservative description of the image. The low-resolution regions are then expanded using the full-resolution image. No new regions are created at full resolution.

The region segmentation algorithm should produce a reliable description of a scene from a range image. The range measurements are corrupted by sensor noise (Section 2.1) which may produce gross errors in the segmentation. The first source of error is the sensor accuracy which degrades rapidly as the measurements are taken further away from the sensor. Due to the limited sensor accuracy, it is difficult to separate regions whose differences in orientation are of the order of the sensor noise. The second source of error is the presence of mixed points at the occluding edges of objects. This problem may lead to erroneous segmentation of the regions that border an object, as well as errors in the estimation of the parameters of those regions.

Merging informations from the reflectance images with the pure range image segmentation removes both types of segmentation errors. Specifically, we are interested in using the edges from the reflectance image. The edges correspond either to occluding edges or to edges on the surface of the object. In the first case, the reflectance edges indicate the possible locations of mixed points, which can therefore be removed from the range image prior to segmentation. In the second case, the reflectance edges may correspond to boundaries between surface patches that may not be distinguishable in the range image due to sensor noise. In the low-resolution segmentation step, pixels that correspond to a window that contains at least one edge pixel are removed so that mixed points at the occluding edges are removed. In the full-resolution step, regions are expanded so that they do not cross an edge. As a side effect, edge pixels are all part of the regions boundaries.

As an example, Figure 15 shows the edges extracted from the reflectance image of Figure 14. The edges were extracted by using a 10×10 Canny edge detector [2]. Figure 16 shows the corresponding low resolution segmentation for a reduction factor of $n = 2$. Each region is displayed with a different gray level. Figure 17 shows the final segmentation obtained at full resolution.



Figure 14: Range and reflectance images



Figure 15: Edges from reflectance image.



Figure 16: Low-resolution segmentation ($n = 2$)

4.3 Object recognition from range and reflectance images

The 3-D features extracted from the range and reflectance images are matched against stored models in order to recognize known objects in the scene. The models are described by a set of surface patches and constraints between them. The constraints encapsulated geometrical properties of the object such as "these two patches are roughly orthogonal", for example. The constraints are implemented as numerical tests on the parameters of the regions extracted from the images. Instead of using strict constraints, such as "the normals v_1 and v_2 of those two regions are exactly orthogonal", we use intervals of confidence within which a given constraint is satisfied, such as "the angle between v_1 and v_2 must be within the interval $[\alpha_1, \alpha_2]$ ". Using intervals allows us to take into account the imprecision on the parameters of the features, and the fact that the stored model may not correspond exactly to the observed object.

The matching between scene and model features first generates hypothesis for each scene feature, and then explores the set of hypothesis in order to find matchings that satisfy the constraints stored in the model. The final product of the matching algorithm is a set of interpretations, that is a set of possible positions of the object in the scene. The interpretations are weighted by comparing the projection of the model onto the range image at the computed location, and the actual observed scene. The interpretation with the largest correlation is retained as the final interpretation (See [5] for a complete description of the recognition algorithm). Figures 18 and 19 show two examples of an object recognized in a range image (in this case a car). The top image is the reflectance image of the scene, the middle image shows the computed location of the car in the range image, and



Figure 17: Final segmentation

the bottom part of the Figures shows an overhead view of the scene with the object superimposed at the computed location. These results show that the combination of range and reflectance images provides the necessary features to accurately recognize and locate 3-D objects in outdoor scenes.

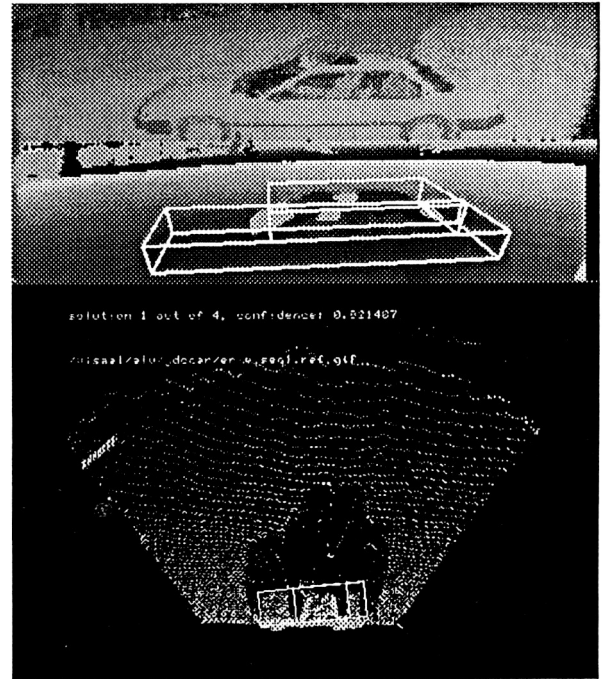


Figure 18: Result of object recognition from range and reflectance images

5 Conclusion

We have developed techniques for the fusion of data from multiple sensors. We have demonstrated the relevance of the resulting merged sensor data in the context of object recognition and scene interpretation for autonomous mobile robots. The experiments with real images showed conclusively that sensor data fusion provides useful additional information for scene interpretation. In order to represent the merged data, we have proposed the concept of a colored-range image in which pixels contain data from different sensors. One obstacle to building colored-range images is the geometric registration between sensors that may take images from different vantage points and with different fields of view. We have found that a simple sensor calibration scheme provides the parameters necessary to perform the registration. Even though we applied the data fusion approach to only three types of data, video, range, and active reflectance images, it is clear that the concept of colored-range image should be extended to other sensors such as sonars, active multiband reflectance, or multiple cameras. The sensor fusion techniques have been successfully integrated into the large autonomous mobile robot systems developed at CMU [10,12], and provide the

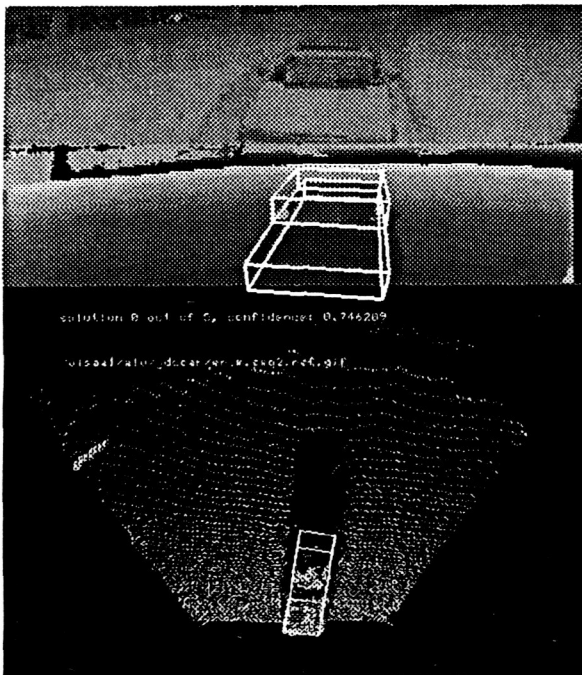


Figure 19: Result of object recognition from range and reflectance images

basis for the development of perception systems for Planetary exploration robots [13].

References

- [1] P. Besl. *Range imaging sensors*. Technical Report GMR-6090, General Motors Research Lab, Warren, MI, March 1988.
- [2] J. Canny. *Finding Edges and Lines in Images*. Master's thesis, Massachusetts Institute of Technology, June 1983.
- [3] O.D. Faugeras M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3), 1986.
- [4] P. J. Besl R. C. Jain. Three-dimensional object recognition. *ACM Comp. Surveys*, 17(1), march 1985.
- [5] M. Hebert T. Kanade. 3-d vision for outdoor navigation by an autonomous vehicle. In *Proc. Image Understanding Workshop*, Cambridge, 1988.
- [6] M. Hebert T. Kanade. First results on outdoor scene analysis. In *Proc. IEEE Robotics and Automation*, San Francisco, 1985.
- [7] D. Zuk F. Pont R. Franklin V. Larrowe. *A system for autonomous land navigation*. Technical Report IR-85-540, Environmental Research Institute of Michigan, Ann Arbor MI, 1985.
- [8] D.G. Lowe. Solving for the parameters of object models from image descriptions. In *ARPA Image Understanding Workshop*, 1980.
- [9] H.P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Technical Report CMU-RI-TR-3, Carnegie-Mellon University, 1980.
- [10] Y. Goto K. Matsuzaki I. Kweon T. Obatake. Cmu sidewalk navigation system: a blackboard-based outdoor navigation system using sensor fusion with colored-range images. In *Proc. First Joint Computer Conference*, Dallas, 1986.
- [11] Y. Ohta. *Knowledge-based Interpretation of Outdoor Natural Color Scenes*. Pittman Publishing, Inc., 1984.
- [12] C.E. Thorpe M. Hebert T. Kanade S.A. Shafer. Vision and navigation for the carnegie-mellon navlab. *PAMI*, 10(3), 1988.
- [13] J. Bares W. Whittaker. Configuration of an autonomous robot for mars exploration. In *Proc. World Conference on Robotics*, Pittsburgh, PA, 1988.
- [14] R. Watts F. Pont D. Zuk. *Characterization of the ERIM/ALV sensor - range and reflectance*. Technical Report , Environmental Research Institute of Michigan, Ann Arbor, MI, 1987.

ROBOT PATH PLANNING USING A GENETIC ALGORITHM

Timothy F. Cleghorn
Paul T. Baffes
Lui Wang

Technology Development and Applications Branch/FM7
Mission Planning and Analysis Division
NASA/Johnson Space Center
Houston, Texas 77058

Abstract:

Robot path planning can refer either to a mobile vehicle such as a Mars Rover, or to an end effector on an arm moving through a cluttered workspace. In both instances there may exist many solutions, some of which are better than others, either in terms of distance traversed, energy expended, or joint angle or reach capabilities. A path planning program has been developed based upon genetic algorithm. This program assumes global knowledge of the terrain or workspace, and provides a family of "good" paths between the initial and final points.

Initially, a set of valid random paths are constructed. Successive generation of valid paths are obtained using one of several possible reproduction strategies, similar to those found in biological communities. A fitness function is defined to describe the "goodness" of the path; in this case including length, slope, and obstacle avoidance considerations.

It was found that with some reproduction strategies, the average value of the fitness function improved for successive generations, and that by saving the best paths of each generation, one could quite rapidly obtain a collection of "good" candidate solutions.

Introduction:

Robotics operations in the 1990's and beyond will be characterized by increasing levels of system autonomy. This implies that the various algorithms being developed now under the guise of Artificial Intelligence will be applied to those robotics operations currently performed or controlled by human operators. It has become traditional to classify robotics operations according to such labels as: manipulation, path planning, sensors, end effectors, etc.; and to deal with these sub-areas independently. However, they are clearly linked, and in some cases solution approaches in one sub-area can be applied to problems in another. A characteristic found in many areas is that there exists a plethora of solutions, and that the problem then reduces to how to distinguish the good solutions from the not-so-good ones.

The robot path planning problem is such a case; and in fact the path can refer either to that of a mobile vehicle such as a Mars Rover or arm platform on the space station (MRMS), or to an end effector on an arm moving through a cluttered workspace. There often exist a large, (even infinite), number of paths between the initial and final positions, and the desire is not necessarily to determine the best solution, but to obtain a "good" one within some reasonable time frame. In the present discussion, we will assume a "global" approach; that is, the space through which the robot moves is fully modeled. The obstructions are assumed to be known, and initially are assumed

to be static. The case of dynamic obstacles will be discussed in a later paper.

There are a number of approaches to this problem, most of which attempt to identify the absolute best path. The problems with these are that the computational times become excessively long, and that "best" is often a very subjective quality. What we have attempted to do is to locate a class of "good" solutions, from which one can be selected arbitrarily. The method we used was what has become known as the "Genetic Algorithm", [1].

The Genetic Algorithm (GA) is based upon fairly simple biological/evolutionary principles. A random initial population is generated, and allowed to evolve in such a way that the desirable characteristics of individuals are on the average enhanced, and the undesirable characteristics suppressed. The characteristics of any individual are described as a bit string, or concatenated bit strings, much as chromosomal material is composed of series of genes. Associated with each string is some "fitness" value which manifests itself as part of the overall fitness of the individual within that particular population. If the strings with high fitness values are retained, and combined with other strings of high fitness, while those of low fitness are allowed to disappear, the average overall fitness of the population should increase.

The mechanism by which the high fitness information is transferred is called "genetic crossover". As in biology, when two individuals mate, the offspring carry some genetic material from each parent. This is called crossover, and can be illustrated as follows:

Parent 1: 101101 0111	child 1: 1011010010
X	and/or
Parent 2: 100011 0010	child 2: 1000110111

Thus a single crossover can produce two new individuals with genetic characteristics similar to their parents. Multiple crossovers obviously can produce many distinct children, but for the present application this was not found to be necessary. What did turn out to be necessary was the inclusion of an environmental driver, specifically, mutation. In simplest terms, this means that any bit has a low but non-infinitesimal chance of being flipped; a 1 becomes 0 and vice-versus. In our application, a mutation had profound and often catastrophic effects upon the survival of the individual, similar to what is found in many natural mutations.

It is necessary to emphasize that the biological analog has limitations, and in one very important respect, we rewrote high-school biology. Most previous work on genetic Algorithms have used bit-strings which were of constant length. Ours not

only varied, but the variation itself was used as the fitness function, and hence as the driver for our "natural selection" law. A second major departure from previous work was the use of "Deification", the process by which we saved the best individual or individuals of each generation through subsequent generations; exempting them from mutation but allowing them to participate in the genetic crossover process. We did, as shall be explained in detail, retain a probabilistic approach: even the individual with the best fitness value can suffer an accident and hence fail to reproduce; although most of the time, the "good" genetic material was maintained and developed.

The Algorithm:

The following terms and definition will be used to describe the genetic algorithm.

Population:	The set of valid paths (solutions)
Generation:	The subset of the population which is under consideration at a given point in time.
Individual:	A single member of the population; in this case, a single valid path. The individual is represented as a bit string, or possibly an ordered list, eg: 1011001011.
Fitness:	Those properties by which the value of the individual is measured. Examples: total path length compared to other individuals, avoidance of steep inclines, energy expenditure, etc.

In order to perform the path-planning task for a mobile robot, the following general approach was taken. First, the terrain map was described. Then an initial group of valid paths were formed, using a random path generator. These paths were treated as the initial generation, and one of several strategies for reproduction was chosen. Successive generations were formed using the reproduction strategy; the process terminating after some arbitrary number of generations, usually 50. This either allowed for convergence to a set of "good" paths, or provided an indication that the strategy chosen was not convergent, i.e., the average and best fitness values failed to improve with time. During the run, the best path solutions in each generation were stored, allowing the selection of the best individual for the total population, even if it did not happen to appear in the final generation. This selection was made simply by choosing that individual path with the best fitness value.

Figure 1 illustrates one of the terrain maps. The black squares represent regions of exclusion, such as boulders. Paths were constructed as moves between adjacent white squares, either laterally or diagonally. It was permissible for the robot to pass between the corners of diagonally adjacent obstacles, but not between their laterally adjacent sides. The number within each square represents the "elevation"; so hills, valleys, craters, and canyons could be represented as well. Cliffs were represented as obstacles, because it was assumed that they were unclimbable, and that falling off of one would terminate the path. The numbers of rows and columns could be adjusted, so arbitrarily accurate maps could be drawn, depending upon the patience of the user.

It should be mentioned, however, that autonomous vehicles generally do not have true global information, but can "see" obstacles only within their immediate vicinity. Thus "global" means limited to the field of view. This does not

invalidate the method for larger areas; it simply requires that intermediate goals be established, and that the algorithm be rerun until the final goal is attained. Except for pathological cases, back tracking should not be required to extend past the preceding sub-goal's origin. Consequently, a grid of about 15X15 was chosen as a compromise between computer time, user input time, and the desire for fineness of detail.

The initial set of paths for each run were constructed by a random path generator. Each initial path had to be valid; that is, it had to start at the user-selected origin, and terminate at the goal, and could not traverse obstacles. Otherwise, loops, hill climbs, crater traversals, etc., were allowed. The number of initial paths was set by the user.

Each valid path, or individual, had a fitness value, which determined that individual's status within its generation, which in turn determined how likely that individual was to reproduce, thus passing the "good" information on to the next generation. The fitness value was determined by: 1) length - the longer the path, the poorer the value; and 2) "energy" expended. The latter was modeled with a hiker in mind; going straight up a steep slope is much more difficult than is going up a switchback. To encourage the vehicle to take a gentle but longer slope up a hill, as opposed to a short but steep climb, a penalty function was devised to be the square of the slope between adjacent points. Figure 2 illustrates two paths to the same goal, and shows their respective fitness values.

Traversals of descending slopes, (not cliffs), required no special energy outlay, so they were not penalized in the fitness values.

Therefore, a number was assigned to each of the initial paths, based upon that path's length and upon its energy efficiency. It should be reemphasized that only valid paths were permitted. The next step was to form a new generation, the average fitness value of which was hopefully smaller (i.e., better) than the original generation's average fitness value. To do this, required selection of one of several possible strategies. Attempts were made to pattern these after zoological analogues, with overlays from various societies, real or mythological. The choice of strategy determined the number of individuals permitted to mate, the number to be culled, the number to be "deified", - as well as the environmental pressure, the rate of mutation.

Mating pair selections and cullings were performed by using a weighted random selection. A "line" was formed, composed of the concatenated fitness values of the individuals within the generation. The line segments were normalized to unity, the value for the individual with the best fitness within that generation. Thus a worse path has a lower numerical value. The "line" is the summation of these segments. If it were intended that 10% of the generation should be culled, a number of "darts" equal to 90% of the generation size would be thrown at the "line". The longest segments have the greatest chance of being hit by a dart; that is, they are selected for survival. Of course, it is possible to miss the "best" in the generation - this happens in nature - the lead elk slips and fractures a leg, or runs afoul of a predator. On the average, however, the least fit of the generation will be eliminated. Subsequently, mating pairs are formed. A "survivors line" is formed in a similar manner, and the "darts" thrown again. Mating pairs are established by pairs of dart throws. As a result, on the average, the best fit will be paired together, although again there is a random element.

Two additional points should be made here. First, it was required that two individuals be selected. If the line segment

representing a single individual was chosen twice within the same mating pair, another "dart" was thrown, until a second individual was obtained. Second, and of great importance for the correct functioning of the G.A., an individual could be selected for more than one mating pair. This is the scheme by which the "good" individuals produce more offspring on the average.

The mating of two individuals leads to genetic crossovers in their offspring. In each case, both possibilities formed from a single crossover were enrolled into the next generation. In practice, when two individuals were selected, a search was made for a common grid point. If several points were found in each individual, one would be selected at random, and the initial portion of each path would be concatenated with the trailing portion of the other. If no common points were found, the individuals would be enrolled as is in the new generation. Thus, in either case, two individuals were added to the next generation. The number of pairs to mate determines the number of offsprings, so the overall size of each generation will shrink, expand, or stay the same according to the strategy chosen by the user. The usual practice was to keep the size of the generation constant, in order to prevent either premature extinction prior to the evolution of "good" paths, or the overloading of the computer by too many individuals. This was done by "filling" the subsequent generation with members of the mating pool, based again on a random weighted selection process.

An additional ingredient was found to be necessary in order to prevent what resembled a collapse of genetic diversity. If a small, but finite mutation rate was not added, after several generations a small group of identical paths was formed. Our "mutation" kept the pot boiling, so that new solutions could be obtained. It also occasionally destroyed a "good" path, which necessitated our saving the best individuals from each generation, as mentioned previously. Mutation was implemented as follows: the mutation probability μ determined whether it would occur within a given individual. If it did occur, a point along the path was selected at random. The remainder of the unmutated path between that point and the goal was destroyed. A new link to an allowed square was made, and the remainder of the path constructed using the random path generator. Clearly, most mutations had catastrophic effects for the fitness value of the mutant, however it did permit new material to evolve, which was not present in the original generation.

A final technique was instigated to accelerate the rate at which "good" paths evolved. The best member, (or members), of each generation is preserved intact for subsequent generations. Deified individuals are free from the threat of mutation, but do remain within the mating pool. The result of this non-biological device is a more rapid convergence toward the "good" solutions, and in the presence of high mutations rates, the necessary ingredient for any convergence.

Experimental Procedure:

The Genetic Algorithm program was developed on a Symbolics 3670, using Zeta LISP. Because the Genetic Algorithm is based upon manipulation of ordered lists, LISP is eminently suitable for this task. For the initial version of the program, a grid board was laid out, upon which the user could define obstacles, as well as elevations.

Following the selection of a specific board, or the construction thereof using the interactive graphics routines available upon the Symbolics, the initial conditions and

parameters for the run were selected. These included population size, number of matings/per generation, number of culled individuals, number of "saved" individuals, and mutation rate; as well as the number of generations to be run. The run was then performed, recording the following information: the best path for each generation, the fitness value for that path, the average fitness for all paths of each generation, and the diversity of each generation. These data could be displayed graphically, as illustrated in Figure 3.

A series of runs were made, using a variety of boards and initial conditions. These could be grouped into three major classes: "Elitist", "Universalist", and "Radio-chemical Wastedump". The Elitist strategy permitted only the most fit members of the generation to mate, as is characterized by high cull numbers and low mating number. It should be recalled that the individuals paired for mating are obtained by a weighted random selection process, and therefore no guarantee exists that the ones with the best fitness will be chosen. Not surprisingly, the number of generations (and hence computer time) necessary to obtain the good paths was greater for this strategy.

An improvement in the convergence was observed by using the "Universalist" strategy: high mating numbers, (usually involving all individuals within each generation), and low cull numbers. Even with the pairing of very good and very bad solutions, the average fitness improved more rapidly than in the case of the "Elitist" strategy.

The "Radio-chemical Wastedump" runs were characterized by high mutation rates. The high rate was imposed upon both the Elitist and Universalist strategies, with the result that convergence to a good solution disappeared. The average fitness values for these runs generally showed no improvement with increasing generation number. It was possible, however, to force improvement by imposing "deification" upon the best individual(s) from each generation; and when this is done, mutation behaves like genetic crossover. The average and best fitness values do improve with time, when both conditions are applied.

It is clear from the latter, that by imposing deification even without the high mutation rate, the convergence would be improved as well; and this was done to decrease the amount of computer time.

Results:

In order to determine the characteristics of the Path Planner, a series of runs were made for increasing complex terrain maps. For each terrain map, one or more of the following input parameters was varied: population size, number of mating pairs, cull number, mutation probability, and number of deified individuals per generation. The output from each set was examined to determine relative convergence rates for good solutions, and how "good" the solutions were compared to the "best" solution for that terrain map.

The initial set was run using a 10X10 square board, (Figure 4) with no variation in elevation. The start position was square (0, 0) and the goal was square (9, 9). The best path had a length (fitness value) of 13.899, and of the thirteen runs using this map, seven found this particular path, and the remaining six had fitness values less than 16.8. The ratio of mating pairs to population size was varied from 0.05 to 0.50. The variation in the convergence to good solutions between these runs was of the same magnitude as that observed in repeated runs using the same input set. Therefore, although there was a slight

indication that a ratio between 0.3 and 0.4 produced better convergence, it could not be isolated from the variation produced just by the random number generator.

In all runs in this set, the mutation probability was held at 0.05, and there were no individuals deified. Rapid convergence to good solutions were seen throughout; in a majority of cases, the best solutions was obtained. No definite trend was observed between Elitist and Universal mating strategies.

Two additional runs were made at this time, examining separate issues. These runs utilized a simple map, consisting of a 10X10 square board with a barricade between the start and goal. The first of these barricade runs was to examine the rate of convergence given a much larger set of possible initial paths. The concern was that good solutions would be much harder to obtain because the initial paths could be much more complex with the additional open spaces. This was found not to be the case. Convergence to good solutions occurred rapidly.

The second barricade run involved raising the mutation probability to 0.65, without saving any individuals. The average fitness and best fitness value for each generation failed to improve. It was found subsequently, that with a high mutation probability it was absolutely necessary to deify one or more individuals in each generation, in order for the best fitness value to improve; and even then the average fitness values failed to converge.

Following these runs, a more complex map was introduced, (Figure 5). This map consisted of 15X15 squares, with scattered obstacles, but still no variation in elevation. A total of eight runs were performed using this map. Again, the major issue was to determine whether there was any significant variation in rate of convergence to good solutions as the ratio between number of mating pairs and population size was varied. Two runs also involved adjusting the mutation probability.

The absolute best fitness (shortest path) for this map was calculated to be 20.97, and results for the eight runs varied between 21.5 and 32.5. Clearly, the results were not as satisfactory with this set as they were with the simpler map. However, convergence was observed in all cases, including that which consisted of mutation probability of 0.90. In fact, this case, in which a single individual was saved in each generation, recorded the best solution for the entire set, although the average fitness did not improve during the entire run. Again, no significant differences were observed between Elitist and Universal mating strategies.

A final run was made with this map utilizing a very low mutation probability ($\mu = 0.0001$). It was observed that the best solution converged fairly rapidly to 23.3, or about 15% above the absolute best value. However, the diversity collapsed to near zero midway through the run, and consequently no further improvement could occur after that point. This illustrates the necessity of a small but finite mutation probability for the successful operation of the algorithm.

The next stage consisted of adding topology into the calculation of the fitness value, and illustrating it on the Symbolics Computer. The small number in the upper left hand corner of each square in Figure 6a represents the "elevation" of that square. Contour lines have been drawn in Figure 6b for better visualization. Represented are: a steep rising slope, a hill, a hole (or crater), and a ditch or "wash". the penalty imposed by climbing up a slope is equal to the square of the local slope. There is no penalty for climbing down.

Thirteen runs were performed using this terrain map. Calculating the absolute best path was extremely difficult; in fact, what is believed to be the best path was found by correcting one of the paths located by the genetic algorithm itself. The fitness value of this corrected "best" path is 23.314, and the best fitness values obtained during the runs ranged between 25.3 and 43.8, (average = 35.11, 51% above best value). The major area of investigation in this set focused upon the effect of deification upon the convergence to the "best" solution. By saving at least one member of each generation, it is guaranteed that the best fitness value solution is at least no worse during subsequent generations. A significant improvement was observed by the deification of multiple individuals within a generation, up to that point at which the genetic diversity collapsed. The best results were obtained with 5-10 individuals saved out of a generation size of 250. It was also observed that slightly better results were obtained using a mating pair number of 0.3 X generation size, although it is not clear that this is statistically significant.

Conclusions:

We have developed a path planner using a genetic algorithm. The principal differences between our algorithm and other genetic algorithms are: 1) the variable length of the list, 2) the way in which we performed crossover and mutation operations, and 3) the use of deification. It should be pointed out that a "greedy Algorithm" was available for post-processing, to straighten out kinks in the final paths, if so desired. None of the results discussed in this paper included that technique. In a real world case, such as onboard an autonomous planetary surface vehicle, this type of post-processing would doubtlessly be employed.

We make the following conclusion based upon our data:

- 1) The genetic Algorithm can be used to construct a robust path planner. Convergence to good solutions were obtained for a wide range of input parameters.
- 2) The inclusion of Deification improves the performance of the algorithm significantly.
- 3) Deification is required for convergence when using high mutation probabilities.
- 4) There is an indication that the optimum ratio of mating pairs to population size lies between the Elitist and Universal mating strategies. This value appears to be in the range 0.3 - 0.4, which implies up to 80% of the population involved in the genetic crossover operation.

It should finally be observed that one of the major drawbacks of the A* algorithm, which is used in many of the existing path planning programs, is that the search-tree is exponential. The corresponding search tree used in this Genetic Algorithm is of order $N \times P$, where N is the number of nodes in the list, and P is the population size. This implies that the Genetic Algorithm is a far less computation intensive approach to path planning.

Reference:

- [1] Goldberg, David E. "The Genetic Algorithm Approach: Why, How, and What Next", ADAPTIVE AND LEARNING SYSTEMS, Plenum Publishing Corp., 1986, pp 247-253

ORIGINAL PAGE IS
OF POOR QUALITY

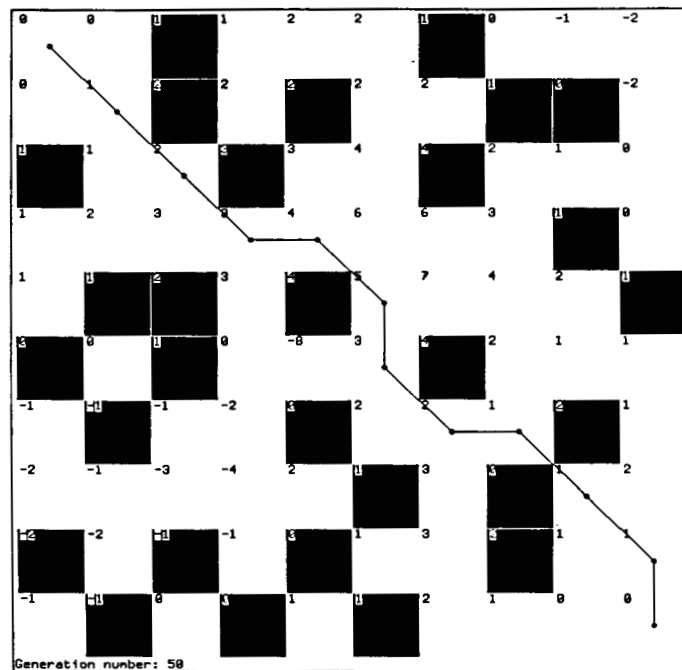


Figure 1. A 10 x 10 terrain map, illustrating obstacles, elevations, and a typical path

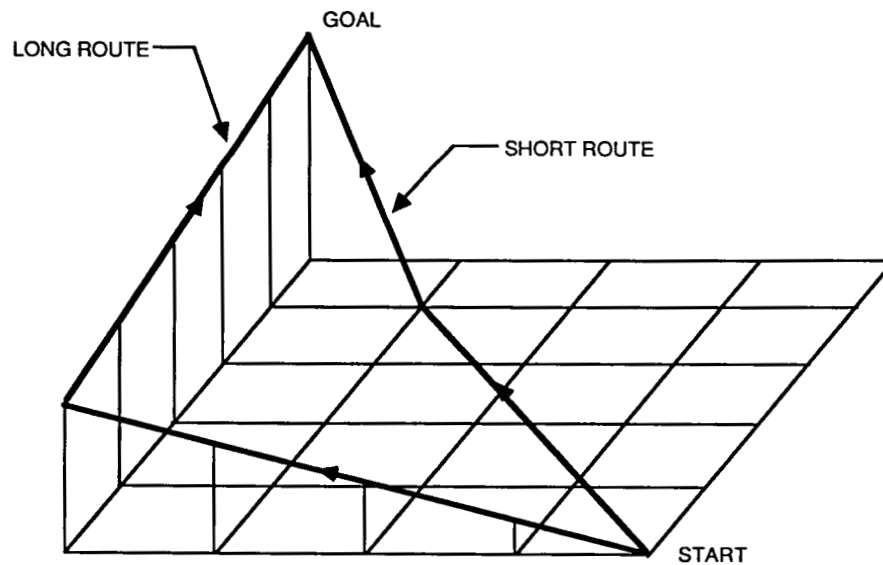
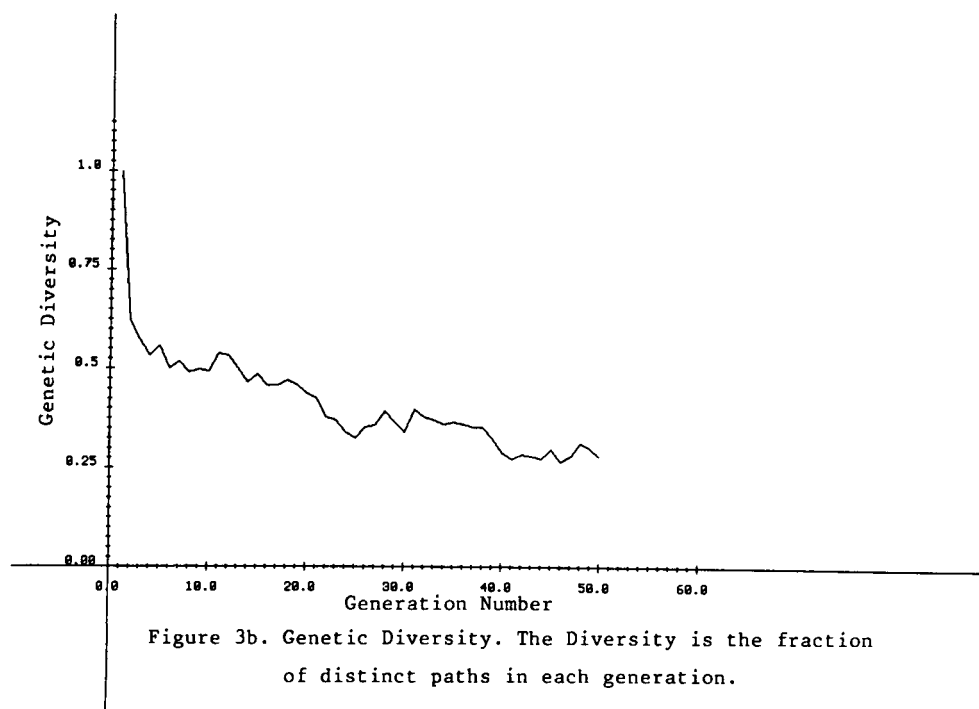
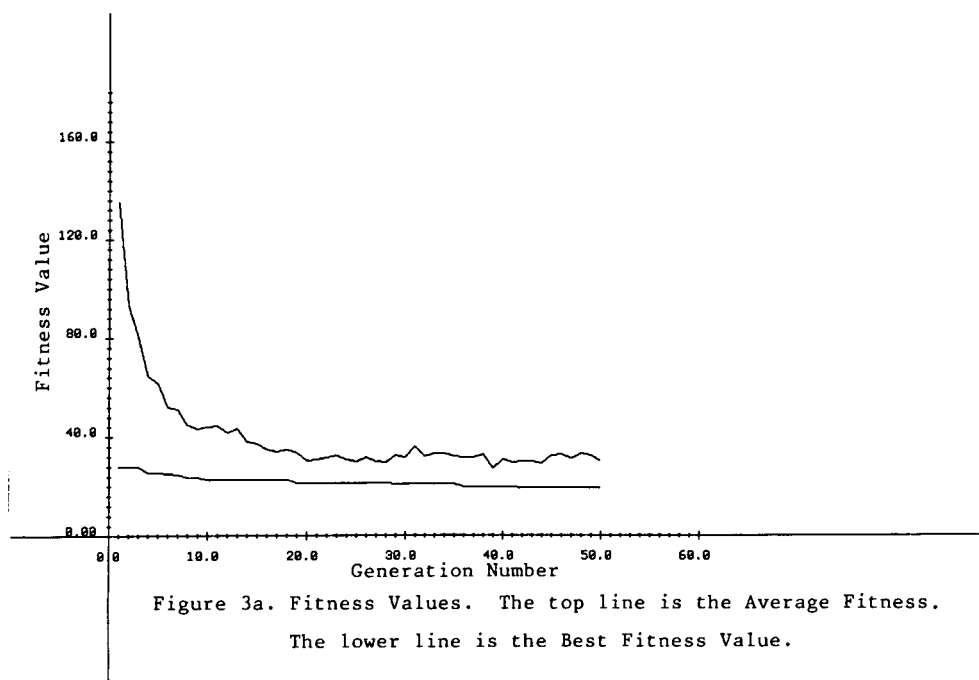


Figure 2. Paths illustrating gradual and steep slopes



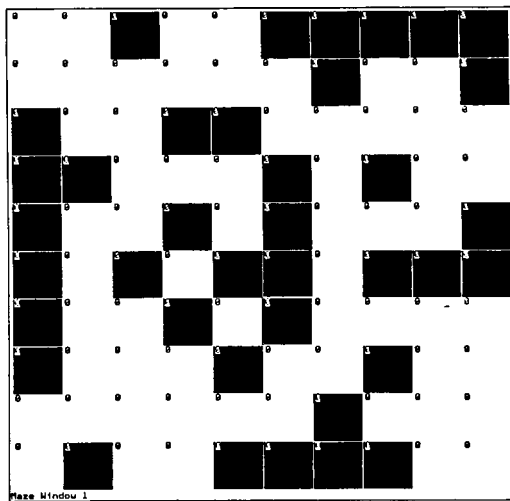


Figure 4. A flat 10 x 10 obstacle map

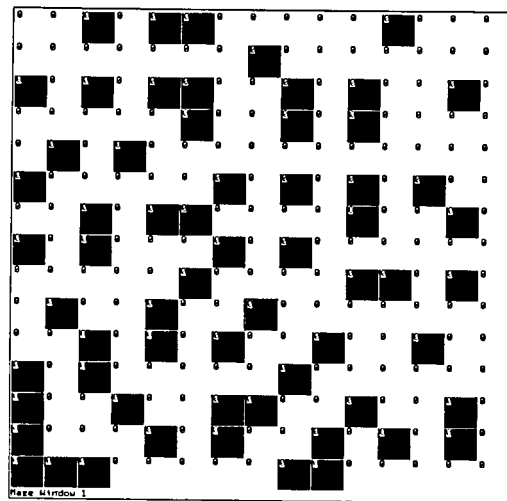


Figure 5. A flat 15 x 15 complex obstacle map

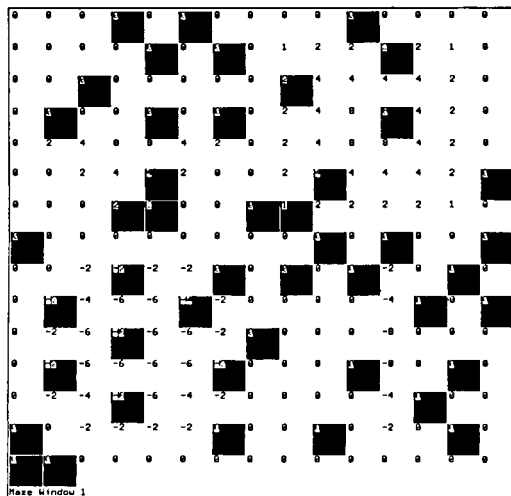


Figure 6a. The 15 x 15 complex terrain map, showing both positive and negative topology.

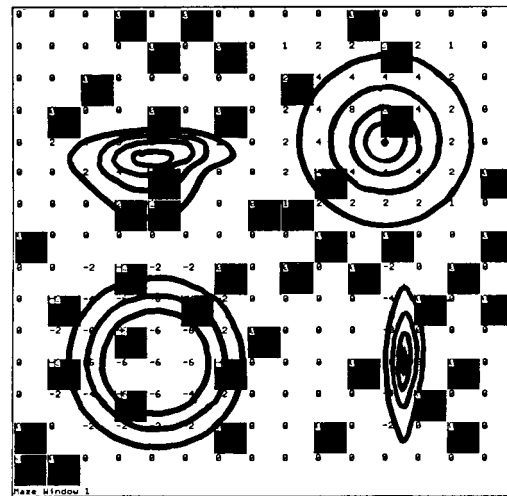


Figure 6b. The same complex terrain map with contour lines at 2 unit intervals.

DESIGN CONCEPT FOR THE FLIGHT TELEROBOTIC SERVICER (FTS)

J. F. Andary, S. W. Hinkal, and J. G. Watzin
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771, U.S.A.

ABSTRACT

NASA has just completed an In-house Phase B Study (one of three studies) for the preliminary definition of a teleoperated robotic device that will be used on the National Space Transportation System (NSTS) and the Space Station to assist the astronauts in the performance of assembly, maintenance, servicing, and inspection tasks.

This device, the Flight Telerobotic Servicer (FTS), will become a permanent element on the Space Station. Although it is primarily a teleoperated device, the FTS is being designed to grow and evolve to higher states of autonomy. Eventually, it will be capable of working from the Orbital Maneuvering Vehicle (OMV) to service free-flying spacecraft at great distances from the Space Station. A version of the FTS could also be resident on the large space platforms that are part of the Space Station Program.

INTRODUCTION

The In-house Phase B Study helped NASA understand operational concepts and scenarios for the FTS. The results will not be used as the design concept for the FTS. Grumman Space Systems in Bethpage, NY, and Martin Marietta, Denver Aerospace in Denver, CO, are conducting more in-depth preliminary design studies.

This paper discusses the technical design drivers that the In-house Phase B Study identified as significant in the development of a robotic system for space. The Phase B Study started with the initial requirements of the top-level mission, system, and functional requirements for the FTS [1]. These requirements were developed during a 2-month Phase A Study conducted by NASA during the fall of 1986 [2 and 3].

The output of the Phase B Study will be integrated with the Martin Marietta and Grumman results to refine the requirements for Phases C and D of the FTS Program that are expected to begin in the spring of 1989.

STUDY APPROACH

The Phase B Study started with a detailed analysis of the Space Station tasks described in the requirements document [1]. These tasks describe generic capabilities that are intended to be representative of the fundamental mission of the

FTS as a robotic device that assists the astronauts in assembly, maintenance, servicing, and inspection tasks in the unpressurized environment of the Space Station.

Analyzing the tasks in the requirements document [1] led to the identification of a number of design drivers for the development of the FTS. These design drivers resulted in a series of trade studies that were used to develop candidate design solutions. The resulting design concept for the FTS was called the "Tinman." This concept resulted in a robotic system that was adequate for the assigned tasks and could perform the tasks reliably and safely.

Advanced technology items were scrutinized as to their relevance to the performance of the assigned tasks as well as their state of readiness. If an item was not considered necessary, it was not incorporated into the design. Some items were considered appropriate, but their state of readiness made them too high a risk for inclusion into the initial implementation of the FTS. High-technology should not be used just for the sake of using it, then to have it fail in orbit. An early failure of the FTS would be a great setback for space robotics. Instead of being a useful tool for the astronauts, the FTS would be discarded and the astronauts would turn to another means of accomplishing the tasks.

A program requirement is that the FTS must be capable of growth and evolution. System adaptability is necessary because of the emerging technologies that will be valuable to the program once they have matured. The FTS must be designed from the ground up with the proper "hooks" and "scars" for growth. With the appropriate systems engineering and architectures that can accommodate growth, advanced technology with software and hardware can be added later to the system with minimum impact. To accomplish this, NASA has adopted a control architecture developed by the National Bureau of Standards (NBS) that permits this type of growth [4].

DESIGN CONCEPT

Figure 1 shows the design concept that was developed for the FTS during the Phase B Study. As shown in the drawing, the telerobot is composed of three major subassemblies: the main body, the manipulator arm assembly, and the arm positioning system.

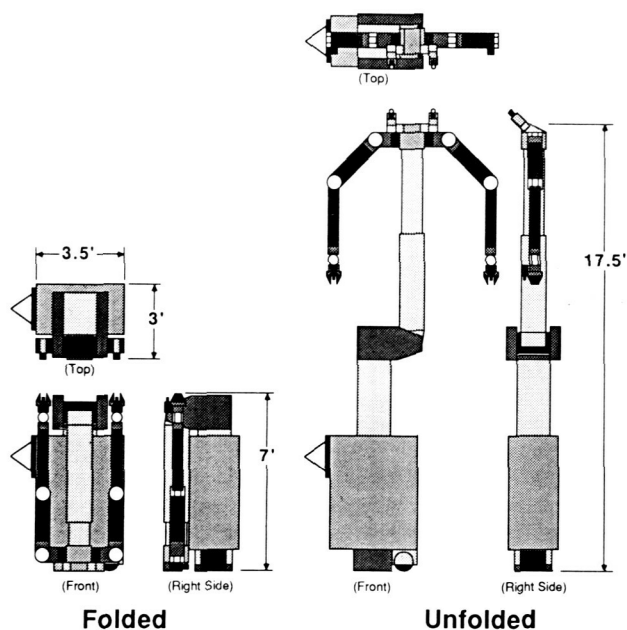


Figure 1. FTS Dimensions

The main body contains all the major electronic components of the telerobot, as well as the grapple fixture by which the telerobot is picked up by one of the large manipulator arms (e.g., the Space Station Remote Manipulator System (SSRMS) or the NSTS Remote Manipulator System (RMS)). The main body also contains the attachment grapple (or foot) by which the telerobot is securely fixed at the worksite.

One of the features of the main body of the telerobot is that it is free to rotate about its central core and the attachment foot. This freedom to rotate allows the thermal radiators, that cover three sides of the main body, to be oriented for optimum heat rejection at the worksite. Main body rotation with respect to the attachment foot allows the operator of the large manipulator arm (SSRMS or RMS) another degree of freedom to help orient the FTS foot for proper mating to the worksite attachment point.

The next major component of the telerobot is the arm positioning system that consists of two, linearly driven, tubular sections connected through an offset rotational joint. The lower section is free to rotate simultaneously with respect to both the main body and the attachment foot. The manipulator arms are free to rotate ± 180 degrees with respect to the upper section. Five degrees of freedom are obtained to position the arms relative to the telerobot main body and attachment location. There are a number of advantages to the arm positioning system: it extends the reach of the telerobot without extending the length of the manipulator arms; it allows the arms to be positioned squarely to a task so that the teleoperator interfaces with the task in a natural manner; and it allows the telerobot to reach out over large objects which may come between the attachment fixture and the location of the task.

The final component of the telerobot is the manipulator arm assembly that is mounted to the end of the positioning system. It consists of the shoulder assembly that rotates

± 180 degrees about the end of the positioning system, and two, 7-degree of freedom manipulators mounted to each end of the shoulder assembly. The manipulators are 1.524 meters (60 inches) long and are configured with a roll-pitch-roll shoulder, pitch in the elbow, and roll-pitch-roll in the wrist.

In addition to the telerobot, the FTS includes two workstation designs: a stowable workstation for the NSTS that is mounted in the aft flight deck of the shuttle and the Space Station workstation that will include FTS unique hardware that will be incorporated into the Space Station Multipurpose Application Console (MPAC).

DESIGN DRIVERS

During the analysis of the requirements and task capabilities, the study team identified the following major design drivers for the FTS:

- Thermal Environment
- Independent Operation
- Manipulator Stability and Positioning
- Safety
- Mobility
- Evolution
- One-G Operation
- Human Interface

The impact of each of these design drivers on the final design concept will be discussed in the following paragraphs. Not all of the design drivers are independent. Often, more than one of the drivers affects the design of a particular subsystem therefore a systems approach had to be taken to the trade studies in order to determine the appropriate solution leading to the best overall design concept.

Thermal Environment

The thermal environment created by the vacuum of space introduces unique problems for the FTS in an area that is only a minor concern for terrestrial robots. In space, the only way of dissipating heat is by radiation or conduction. The only paths for conduction were by hookup to the Space Station thermal system or by dumping heat into the FTS base mounting structure. Both options were considered too restrictive for the flexibility and usefulness of the FTS and they also created a thermal interface to the Space Station that the design team wanted to avoid. Therefore, radiation was the only means of heat dissipation.

Radiating heat from a robot with peak operating power in the 1 to 2 kW range with approximately 20 motors, several high-speed computers, video equipment, and batteries with heat dissipation as the only means of cooling resulted in a thermal problem. The operation of the FTS should not be restricted because of the thermal environment. This meant that the FTS had to be capable of operating with arbitrary

sun angles and with partial blockages from the structure at the worksite.

To overcome these problems, the overall power of the telerobot was reduced, its total radiating capability was increased, and the main battery was removed from the telerobot.

One effect of reducing the power was the selection of motors at each joint that were sized for the tasks in zero gravity but could not operate without assistance on Earth. By using smaller motors, the manipulator thermal system could be separated from the rest of the body and it could collect all the other heat dissipating components into one structure that could be optimized for thermal radiation.

Figure 2 shows the concept for the telerobot body that uses heat pipes to direct the heat from the electronic boxes out to the outside surfaces where radiators cover three sides. The main body was designed to rotate independent of the manipulators and the arm positioning system so that it could be controlled to track an optimal orientation to cold space as the telerobot is performing its tasks.

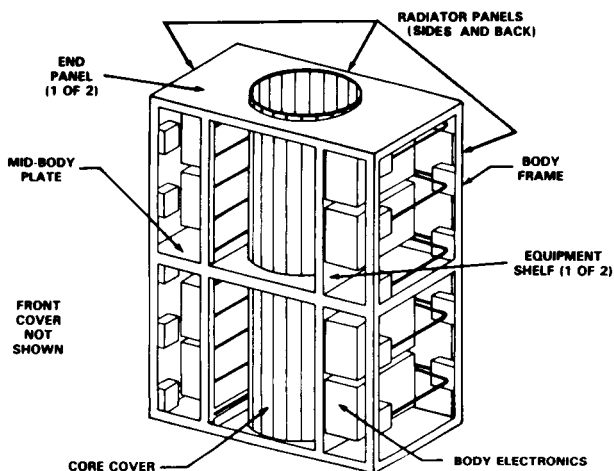


Figure 2. Structure Subsystem Tinman Design

Removing the main battery from the telerobot had a number of effects on the design. It reduced the mass of the telerobot and removed a source of power dissipation. It also freed the telerobot from the tight thermal limits that the battery imposed on the system.

The combined effect of all these design choices produced a thermal design that is independent of the Space Station that will permit indefinite operation of FTS under most conditions. In some extreme cases of radiator blockage, the task may have to be halted temporarily to allow the telerobot to cool down. The use of a small "backpack" was considered composed of Phase Change Material (PCM) that could be used to absorb peak loads to enable the telerobot to continue operating for a brief time under extreme conditions. The thermal system is also an ideal candidate for the incorporation of an expert system that could continually monitor

the thermal health of the telerobot and inform the operator how much time is left before a cool-down period would be required.

Independent Operation

Another requirement is that the FTS must be capable of limited operation independent of hard-wired utilities for power, data, and video from the Space Station. As a result of this requirement, a large battery and an RF communications system was included in the design of the FTS. The FTS can never be totally independent of the Space Station because it always needs a firm structural attachment when working. However, the requirement for independent operation gives the FTS a tremendous amount of flexibility allowing it to work in areas on the Space Station where no utility ports are located.

A battery that would allow operation for even a few hours at the power levels of the FTS adds considerable weight and adversely impacts the thermal subsystem. Since the independent operation is not the primary mode of operation, it was decided to remove the battery and the communications system from the main body of the telerobot and locate them in a separate module called the Robot Support Module (RSM). Because there is not a requirement for an early independent operational capability, the RSM could be launched later than the FTS thereby reducing the initial manifested weight of the FTS.

Another advantage of the separate RSM is that it would be possible to design different RSMs for the different operating environments of the FTS. The NSTS and the Space Station have different power and communications systems, therefore a different RSM could be designed for each location. Another RSM could be built for operation from the Orbital Maneuvering Vehicle (OMV) for the servicing of free-flying spacecraft away from the NSTS orbiter or the Station as shown in Figure 3. Two RSMs on the Space Station itself are a possibility so that while one is being used, the other could be having its battery recharged.

Manipulator Positioning and Stability

When the work environment of the FTS is examined in both the shuttle payload bay and on the Space Station, the same dimension of 5 meters keeps reoccurring. The shuttle payload bay is 4.57 meters wide and, consequently, most payloads launched by the shuttle are also approximately 5 meters wide or 5 meters in diameter. The SS truss bays are 5-meter cubes and the Attached Payload Accommodation Equipment (APAE) sit on a 5- by 5-meter base. It can be concluded from this information that the ideal reach envelope of the telerobot would be 5-meters. If the telerobot is to work in these locations, it must be able to cover these types of distances. However, early analysis indicated that a 5-meter reach for the manipulator arms was not feasible if the telerobot was to do any dexterous manipulation. A local mobility system and an arm positioning system was chosen to deliver the arms to the task. This approach allows the arms to be shorter and more rigid for the fine control tasks.

Figure 4 shows the reach envelope of the telerobot. Situated in the center of the Space Station truss bay, the telerobot

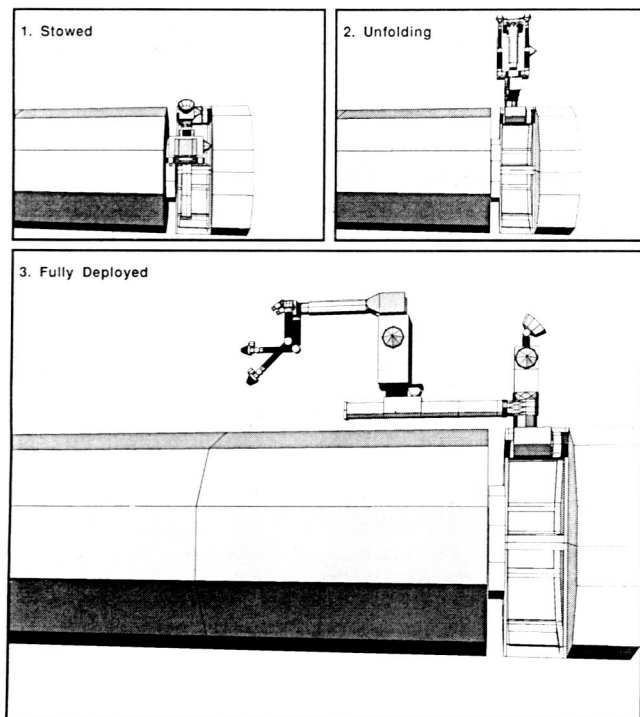


Figure 3. FTS/OMV Servicing

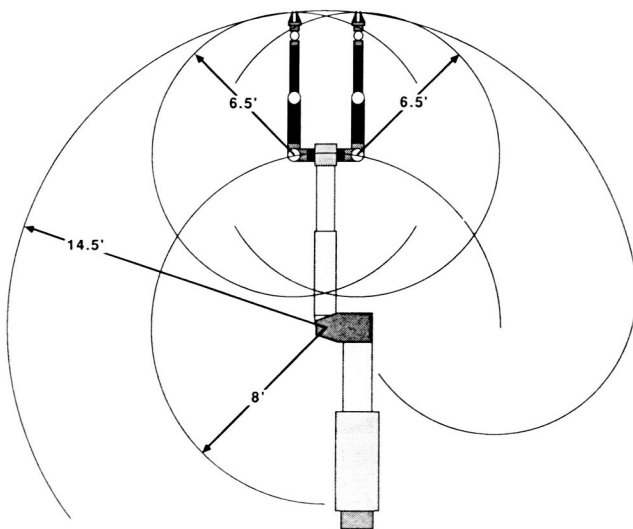


Figure 4. FTS Work Volume

can reach all faces of the bay. The reach of the telerobot at an APAE site is shown in Figure 5 where the Orbital Replaceable Units (ORUs) in the center can be reached from either side, even if larger ORUs are in the way.

The flexibility and controllability of such a system are still areas of concern that are being investigated. Preliminary indications are that the arm positioning system can be made

rigid enough to meet the task requirements. The 5 degrees of freedom in the arm positioning system are controlled open-loop and, therefore, do not contribute complexity to the arm control problem. The degrees of freedom in the positioning system are commanded to set positions one at a time and then rigidly locked before the operator begins to use the manipulator arms. It is not anticipated that the positioning system would be teleoperated through the hand-controllers. The operator could simply key in the position of the joints from a keyboard.

Safety

Safety is of primary importance in the design of the FTS. Safety influences each subsystem and must be designed into the FTS from the start. The Phase B Study approach was to set up a watchdog safety subsystem that consists of redundant radiation hardened computers and associated sensors in the telerobot to monitor all aspects of the telerobot operations and health. Also, the workstation has a safety computer that acts as a global safety monitor for workstation operations as well as the telerobot safety subsystem. Whenever any anomalous condition is detected, the safety computers will stop all movement of the telerobot.

There is also a safety shutdown signal that originates from an astronaut on Extravehicular Activity (EVA) if he senses a problem with the telerobot. This is called the EVA safety link and allows an EVA astronaut to have shutdown control of the telerobot whenever he is working in the vicinity of the telerobot.

Each controller for the manipulator joints is capable of being programmed to limit the local parameters associated with that joint, such as velocity and acceleration. This programming allows the motions of the telerobot to be tailored to the task and the environment. A velocity limit of 1 foot per second is imposed on the manipulators whenever the telerobot is working in the vicinity of an astronaut or critical hardware. Similar limits must be imposed on the maximum momentum the system can attain when moving an object. This may result in an even lower tip velocity, but it ensures that the telerobot can safely brake its motion to avoid collision.

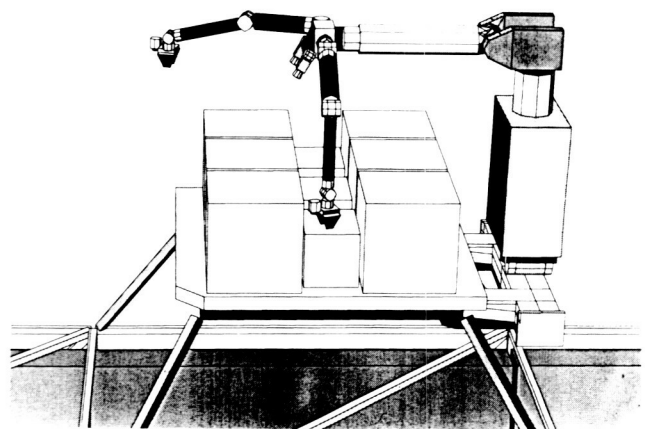


Figure 5. FTS Operating from an APAE

Another safety feature in the telerobot is the inclusion of a small, holdup battery within the telerobot to sustain its functions and to perform an orderly shutdown in the event of a power loss. This safety feature is needed when the telerobot is operating without the large battery in the RSM, and it is deriving its power from the host vehicle.

Mobility

Mobility was identified early as an FTS design driver. There is not a requirement for the type of mobility that would allow the telerobot to walk down the Space Station truss. There are other means available on the shuttle and the Space Station to provide global mobility, such as the RMS on the shuttle and the SSRMS on Space Station attached to a transport device such as the Mobile Servicing Centre (MSC) or the Mobile Transporter (MT). However, from a close examination of the FTS tasks, it is clear that some form of "local mobility" (or "robility") was needed at the worksite in order to make the FTS a useful tool on the Space Station.

The local mobility system that is part of the in-house concept is a portable rail that can ride out to the worksite with the telerobot to provide lateral movement. The portable rail, together with the arm positioning system, allows the manipulator arms to be positioned with 6 degrees of freedom at the worksite. The length of the portable rail had to be traded off against the flexibility of the rail and the induced motions at the end of the rail when the telerobot is in operation. The portable rail is attached to the RSM in the in-house concept so that the telerobot/rail/RSM combination can be picked up as one unit and carried to the worksite by one of the transport devices on Space Station. Figure 6 shows the portable rail supporting the telerobot from the RSM.

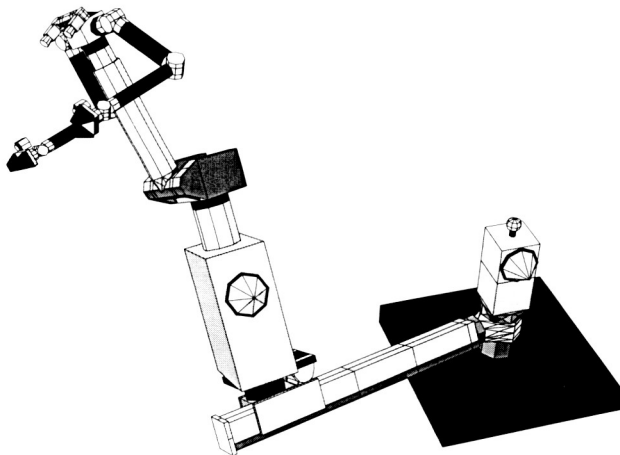


Figure 6. FTS and RMS (Robot Support Module)

Evolution

The FTS must be able to evolve towards greater adaptability which includes more autonomous operation that will be accomplished through the incorporation of advanced hardware and software items as they become available. Since

the FTS is intended for permanent residence on the Space Station, new items must be added to the system in orbit. The FTS must be designed to easily accept these changes. This will be done by the incorporation of modularity and accessibility in the design of all subsystems of the FTS and by a careful implementation of the NASREM architecture.

Primary growth areas are expected to be in more advanced computers, upgraded software, advanced sensors with image processing, smart end effectors, and new and more efficient power systems. Also, the manipulator arms could be of a modular design so that they can be reconfigured to provide more capability for new maintenance and servicing tasks on the Space Station. Power, data, and video lines would run throughout the telerobot with standard interfaces defined at the tool plate, arm joints, and other locations where hardware may be added or later changed.

A vision system, which initially is just a closed circuit video system, can easily grow to a stereo-vision system and eventually evolve to full machine vision. Steps that can be taken in the initial design to facilitate this growth are the choice and location of cameras and the interfaces to permit the computers to have access to image data.

One-G Operation

Requiring that the telerobot exhibit its full operational capability in the gravity environment of Earth, has far reaching impact on the system design. From a programmatic standpoint, the FTS must be capable of being tested in the performance of representative tasks on Earth before it is committed to launch. However, such a requirement has to be weighed against the impact it causes on the structural, controls, electromechanical, power, and thermal subsystems.

For terrestrial robots, a 100:1 weight-to-lift ratio is not unusual, and a ratio of 10:1 is just now being achieved by some research manipulators such as the Laboratory Telerobotic Manipulators (LTM) being developed by NASA Langley Research Center and the Oak Ridge National Laboratory. This means that if the FTS were required to handle mockup hardware weighing 50 pounds, the manipulators would be on the order of 300 to 500 pounds each using today's technology. This results in 600 to 1,000 pounds for just the manipulators. The total manifested weight for the FTS, including the telerobot and the workstation, is presently 1,500 pounds.

The FTS must undergo a strict weight control program that will result in motors and a structure that will be adequate to accelerate the inertias required by the tasks in the zero-gravity environment of space, but they may not be capable of lifting the mockups of the same hardware on Earth. This will mean that the telerobot will need special assistance to perform its operations in 1 G, such as counterweights and other gravity off-loading devices.

Smaller, lightweight motors are a benefit to both the power and thermal subsystems of the FTS. A lighter weight structure has an impact on the control system since the manipulators will be more flexible, but this is not viewed as an insurmountable problem for the FTS because of the recent advances in algorithms for the control of flexible robots.

Human Interface

The design of the FTS for the human operator extends beyond the obvious human engineering of the workstation, (e.g., ensuring that the operator is presented with all the necessary displays and controls). The FTS is a teleoperated device where the operator is directly in the control loop. The human interface has a strong influence on the design of the control system, the data system, and the sensors, including the vision system.

The FTS must be designed for operation by one operator. Inventive means must be found for the control of the cameras, illumination, and other peripheral devices when the operator is using both hands to operate the manipulators.

The study team concluded that the use of force reflecting hand controllers should be a requirement for the FTS. This would permit the operator to sense the manipulator forces in his hand controllers. For a teleoperated device, this requirement is a tremendous asset to the operator. It enhances safety when working in an unstructured environment, and it has been proven through documented experiments in the laboratory to reduce errors and overall training time.

The problem on force reflection is the stringent data latency requirement it places on the data system for communications between the workstation and the telerobot. Because the force loop is now closed through the workstation, the stability of the control loop depends upon minimizing the delay time for the round trip signal. The loop should operate at approximately 200 Hz, which results in a latency requirement of 5 msec. The FTS will be using the Data Management System (DMS) on the Space Station to connect the workstation to the telerobot, and an assessment has to be made to see if the DMS can satisfy such a latency requirement.

CONCLUSION

The Flight Telerobotic Servicer promises to be a useful, reliable, and safe tool to assist the astronauts in performing assembly, maintenance, servicing, and inspection tasks on Space Station and the NSTS. The design challenges have been identified and operational scenarios and task planning have been addressed by the NASA Phase B Study team while candidate designs are being developed by Grumman and Martin Marietta in their Phase B Studies.

The FTS is unique in that it will be required to operate in a much less structured environment than previously developed industrial robots. It will be required to perform

many varied tasks with varying precision throughout its expected lifetime. These tasks will increase in complexity therefore the system must be capable of substantial growth and evolution. It is a program that focuses more on the future than the present technology.

ACKNOWLEDGEMENTS

The information presented in this paper has been obtained from numerous sources. The major contributors were:

Engineers from—Goddard Space Flight Center, Johnson Space Center, the Jet Propulsion Laboratory, Marshall Space Flight Center, Ames Research Center, Langley Research Center, the National Bureau of Standards, and the Oak Ridge National Laboratories.

Universities—Dr. Richard Voltz, University of Michigan; Dr. Pradeep Khosla, Carnegie Mellon University; and Dr. Edward Haug, University of Iowa. Dr. David Criswell, University of California, San Diego, headed a team of FTS researchers from the Consortium for Space and Terrestrial Automation and Robotics (CSTAR) which included: Dr. George Kondraske, University of Texas, Arlington; Dr. Michael Walker, University of Michigan; Dr. Ken Lauderbaugh, Rensselaer Polytechnic Institute; and Dr. Kai-Hsiung Chang, Auburn University.

Computer graphics—Eugene Aronne, ATR.

REFERENCES

- [1] *Flight Telerobotic Servicer Requirements Document for Definition and Preliminary Design*, SS-GSFC-0028, April 1987.
- [2] Andary J., S. Hinkal, and J. Watzin, *The Flight Telerobotic Servicer (FTS): A Focus for Automation and Robotics on the Space Station*, Paper IAF-87-25 presented at the 38th Congress of the International Astronautical Federation, Brighton, U.K., October 10-17, 1987; *Acta Astronautica*, Issue 7/8, Vol. 17, July/August 1988.
- [3] *Flight Telerobotic Servicer Strawman Concept Engineering Report*, SS-GSFC-0031, March 15, 1987.
- [4] Albus J., H. McCain, and R. Lumia, *NASA/National Bureau of Standards (NBS) Standard Reference Model for Telerobot Control System Architecture (NASREM)*, SS-GSFC-0027, December 4, 1986.

THE WCSAR TELEROBOTICS TEST BED

N. Duffie and J. Zik
College of Engineering
University of Wisconsin
Madison, Wisconsin 53706

R. Teeter and T. Crabb
Astronautics Corporation of America
Astronautics Technology Center
Madison, Wisconsin 53716

ABSTRACT

One of the objectives of the Wisconsin Center for Space Automation and Robotics (WCSAR) at the University of Wisconsin-Madison is the development of component technologies for use in telerobotic systems for space. As part of this effort, a test bed has been established in which these technologies can be verified and integrated into telerobotic systems. The facility consists of two slave industrial robots, an articulated master arm controller, a cartesian coordinate master arm controller, and a variety of sensors, displays and stimulators for feedback to human operators. The controller of one of the slave robots remains in its commercial state, while the controller of the other robot has been replaced with a new controller that achieves high-performance in telerobotic operating modes.

A dexterous slave hand which consists of two fingers and a thumb is being developed, along with a number of force-reflecting and non-force reflecting master hands, wrists and arms. A tactile sensing finger tip based on piezo-film technology has been developed, along with tactile stimulators and CAD-based displays for sensory feedback and sensory substitution.

This paper describes the WCSAR telerobotics test bed and the component technologies that are incorporated in it. It also describes their integration of these component technologies into telerobotic systems, and their performance in conjunction with human operators.

INTRODUCTION

Autonomous robots working in industrial environments, typically, replace human workers performing well structured and repetitive tasks. Advanced teleoperated systems, on the

other hand, extend the human manipulation, sensing and cognitive capabilities to remote locations. This shields the operator from the hazards of working in the task environment. The need for use of both autonomous and teleoperated (or a combination of the two) robotic systems for space applications is well recognized by NASA [1] and the aerospace community. Major emphasis in much of this work is placed on developing [2] human-like robotic systems that would replace humans in extra-vehicular activities. This includes a major effort by NASA for the development of the Flight Telerobotic Servicer.

The work underway at the Wisconsin Center for Space Automation and Robotics (WCSAR) is designed to complement such efforts by developing modular and add-on technologies that would improve the effectiveness of such systems as well as enhance effective utilization of automation and robotics in commercial activities in a space environment. The emphasis is on the development of new component and system technologies that will not only utilize, but improve the existing automation and robotic technologies for applications involving assembly, maintenance and servicing of space platforms, stations, commercial satellites and future production and life support facilities. The technology areas of near-term are those that enhance dexterity, sensory perception, performance, and telepresence in telerobotic systems

WCSAR is a NASA-funded Center for Commercial Development of Space (CCDS) and was founded in 1986. It provides an organizational structure that fosters the co-sponsorship of commercial application of space automation and robotics between industrial partners and associated universities. At present, WCSAR has three active project areas: (1) Astrobotics™ - Robotic Technology Development; (2) Astroculture™ - Automated Plant Growth Facilities for Space; and (3) Astrofuel™ - Automated Lunar Resource Processing Systems. These are being pursued by interdisciplinary teams formed by WCSAR's university and industrial partners.

The layout of the WCSAR telerobotics test bed is shown in Figure 1. It consists of three major subsystems:

- 1) **Cincinnati Milacron Robot with Enhanced Control and Dexterous Hand.** This subsystem consists of a Cincinnati Milacron T³-726 electric-drive robot, an ACA dexterous slave hand, a number of master hands, and a non-kinematic replica master arm as illustrated in Figures 2 and 3. The original controller of the robot has been replaced with a new, higher-performance controller designed at WCSAR which is capable of being flexibly programmed in a number of telerobotic operating modes.
- 2) **ASEA Industrial Robot with Telerobotics Inc. Gripper.** This subsystem consists of an ASEA IRB 6/2 electric-drive robot, its controller, a cartesian coordinate master arm, a Telerobotics, Inc. slave gripper, and a force-reflecting master gripper as illustrated in Figures 4 and 5. Control of the ASEA is accomplished with a host computer as the master, serially linked to the ASEA controller via a standard RS232C communication port on the ASEA.
- 3) **Space Shuttle Aft Flight Deck Mock-Up.** High-performance telerobots working in space with sensory feedback to the operator will need to be operated from workstations in space within reasonable communications proximity. One likely workstation location is the aft flight deck of the Shuttle; another is the Space Station. A mock-up of the aft-flight deck of the Space Shuttle is included in the test bed for the purpose of simulating teleoperation in a space environment. It serves to isolate operators of telerobotic experiments from slave hardware, thus insulating them from visual and audio cues that would not be present in space.



Figure 2. "Milacron" subsystem with dexterous slave hand

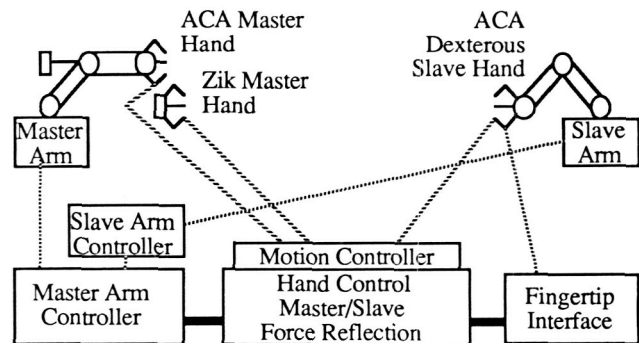


Figure 3. Schematic of "Milacron" master/slave subsystem

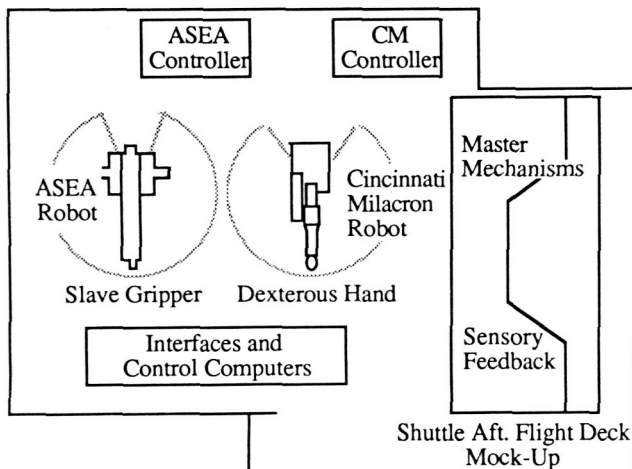


Figure 1. WCSAR Telerobotics Test Bed

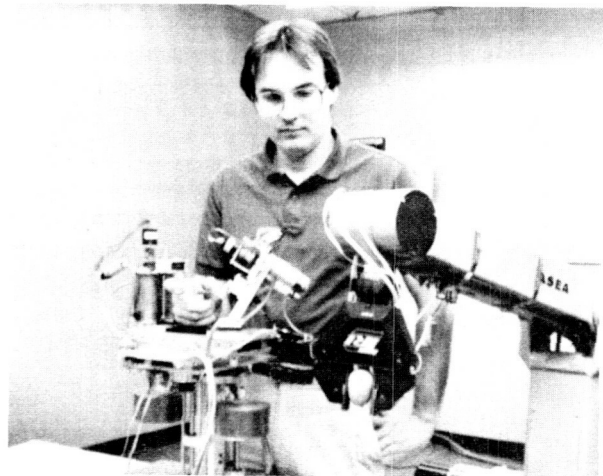


Figure 4. "ASEA" subsystem with master/slave dexterous hand

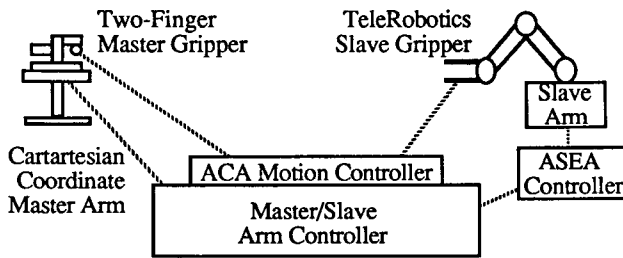


Figure 5. Schematic of "ASEA" master/slave subsystem

MASTER HANDS, WRISTS AND ARMS

Table I lists a number of devices that have been or are presently being developed at WCSAR. The first developments were centered around the "ASEA" subsystem and the master arm and hand shown in Figure 4. With a cartesian master arm interfaced to the robot using a standard communication interface, teleoperation control of the robot was achieved relatively quickly. However, performance in this system is limited by low-speed communications. The cartesian coordinate master provides a large mounting platform to locate the master hand which incorporates drive motors for force reflecting fingers. This setup will allow WCSAR to perform a series of force-reflecting experiments using the parallel motion slave hand.

Next, a master arm and wrist was developed which provides six-degrees-of-freedom for control of the "Milacron" subsystem shown in Figure 2. Also, a number of dexterous master hands have been developed to interface with the dexterous slave hand. Figure 6 shows a light-weight master hand with no force reflection. This device is referred to as "Laird H1" in Table I and features three fingers, each with three degrees of freedom.

Figure 7 shows one finger of a force reflecting master hand that is under development. Its features include:

- a three finger design which allows for object manipulation;
- three degrees-of-freedom per finger which allow for various types of finger manipulation;
- force reflection to enhance telepresence and reduce task completion time;
- low friction and inertia; and
- position range same as a human finger;

The device mounts to a bracket that is fixed to the back of the operator's hand (not shown in Figure 7). Forces are transmitted to the operator's finger through a

tendon/pulley/linkage system [3,4]. Attachment to the finger is realized with a ring that fits around the finger. Force reflection is implemented using four remotely-located motors. Torque from the motors is transmitted directly to the master finger. One finger of the master hand has been manufactured and is currently undergoing testing.

Table I. Master arm/wrist/hand developments

Component	Characteristics	Status
Cartesian Coordinate Arm Master (Zik A1)	<ul style="list-style-type: none"> • Cartesian coordinate configuration with three degrees-of-freedom • Standard industrial interface (RS232 comm. to ASEA) 	Complete
Non-Kinematic Replica Master Hand Type A (Zik H1)	<ul style="list-style-type: none"> • Two digit (finger) device • One degree-of-freedom per digit • Force reflection implemented • Actuators local 	Complete
Non-Kinematic Replica Master Hand Type B (Laird H1)	<ul style="list-style-type: none"> • Three digit (finger) device • Three degrees-of-freedom per digit • Force reflection provided with one drive motor per finger • Actuators local 	Complete
Non-Kinematic Replica Master Hand Type C (Laird H2)	<ul style="list-style-type: none"> • Three digit (finger) device • Three degrees-of-freedom per digit 	Complete
Non-Kinematic Replica Master Hand Type D (Zik H2)	<ul style="list-style-type: none"> • Three digit (finger) device • Three degrees-of-freedom per digit • Force Reflection provided with each degree-of-freedom • Actuators remote 	Currently being manufactured
Non-Kinematic Replica Master Hand Type E (ACA H1)	<ul style="list-style-type: none"> • Three digit (finger) device • Two degrees-of-freedom per digit 	Currently being manufactured
Non-Kinematic Replica Master Arm/Wrist (Zik A2)	<ul style="list-style-type: none"> • Six degrees-of-freedom • Mechanically counterbalanced • Three intersecting axis wrist configuration with hand grip in center of wrist 	Currently being manufactured

THREE-FINGERED DEXTEROUS SLAVE HAND

The Astronautics Corporation of America (ACA) dexterous slave hand is under development in conjunction with WCSAR and the State of Wisconsin and has the following features:

- dexterity is achieved with three digits, eight or nine movable joints, and five to seven independent movements;
- power and speed are similar to the human hand;
- self-contained actuators are in a single mechanical module;
- capability of operating in teleoperation modes with reflected force and compliant control;
- minimal size and weight;
- simplicity and robustness suitable for space and commercial use; and
- precision and controllability.

A prototype of one of the three digits has been fabricated, assembled and tested as shown in Figure 2. The three-fingered slave hand consisting of three fingers and a thumb is illustrated in Figure 8, and its testing in the test bed will begin at the end of July, 1988.

TACTILE SENSORS AND STIMULATORS

Tactile feedback in a telerobotic system can provide the operator with an accurate sense of a presence when the operator is manipulating and contacting objects with the remote slave device. One concept currently being investigated is shown schematically in Figure 9. It consists of three major elements: sensors, displays, and the stimulator interface. The sensors currently used include very small piezoresistive devices that are highly sensitive to pressures of less than 3 psi, and are mounted to the slave in areas where tactile contact is most common and most useful. The sensors will be attached to the parallel jaw grippers on the "ASEA" subsystem and to the dexterous hand on the "Milacron" subsystem to conduct typical EVA tasks in a teleoperated mode.

Because human visual and auditory sensory inputs may be heavily utilized in a telerobotic system during these tasks, tactile stimulation has been the primary display mechanism to the operator. The goal of the system is to provide this stimulation directly to the operator's hand in a pattern matching the sensor pattern on the slave hand. This direct mapping of the slave sensors and the master stimulators will yield a comfortable system with the least amount of operator interpretation and training required.

ROBOTIC FINGERTIP SENSOR

Researchers at WCSAR have adapted a four-degree-of-freedom, tactile fingertip sensor, originally developed for manufacturing applications [5], for use in teleoperated systems. Integration of this fingertip on the dexterous hand will allow WCSAR to develop sensitive force feedback from the slave

robot and enhance telepresence. Eventually, the sensor will allow a teleoperated robot to grasp fragile objects and do delicate work. The fingertip:

- can be optimized with respect to its shape and its sensing element size and location to give favorable signal characteristics;
- is linear over a reasonable range of applied forces;
- has low hysteresis and good repeatability;
- can be linearly decoupled to allow measurement of four degrees-of-freedom appropriate for fingertip grippers. (normal force, two tangential force components, and torque about the normal force axis [6]);
- is inert to most environmental influences;
- can be easily constructed with minimal material costs; and
- can be made rugged, yet possesses sufficient compliance for practical force control.

The basic concept of the sensor is shown in Figure 10. The sensor is constructed of two materials: room temperature vulcanizing (RTV) silicone rubber, which is used as a compliant, homogeneous, bulk base material, and polyvinylidene fluoride (PVDF) film, which is the piezoelectric polymer used to detect strains in the rubber. When a force is applied to the fingertip sensor, deformation in the rubber results. This deformation is transferred to four strips of piezoelectric film which are bonded to the rubber's surface. The shift in electrical charge in the strained piezoelectric film is the signal used to measure the forces applied to the sensor.

Each different component of the force applied to the sensor, whether it is normal, tangential, or torque, will produce a unique signal in each of the four pieces of piezo-film. Therefore, after the signals have been amplified, the signals must be sent to a computer for decoupling which will resolve the applied force into its independent components. The characteristics of the fingertip are listed in Table II.

Table II. Characteristics of fingertip sensor

Linearity	1% Error at 4 Pounds of Normal Force
Hysteresis	3% Error at 7 Pounds of Normal Force
Repeatability	< 1% Error at 5 Pounds of Normal Force
Range	0.5 Ounces to 15 Pounds

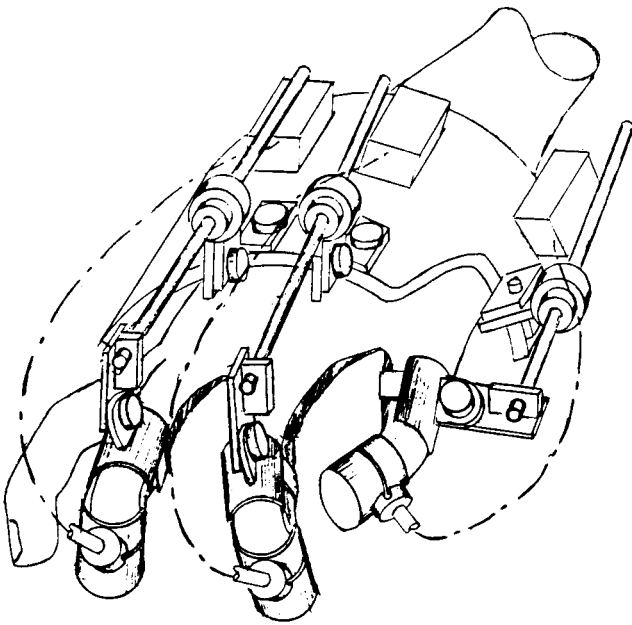


Figure 6. Non force-reflecting master finger

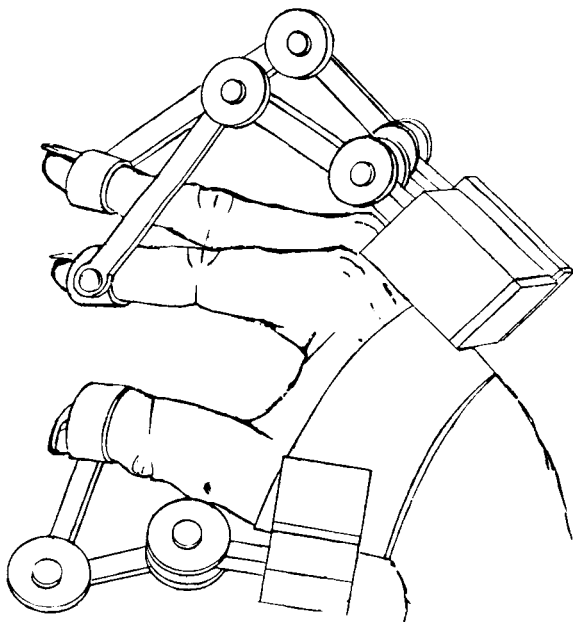


Figure 7. Force-reflecting master finger

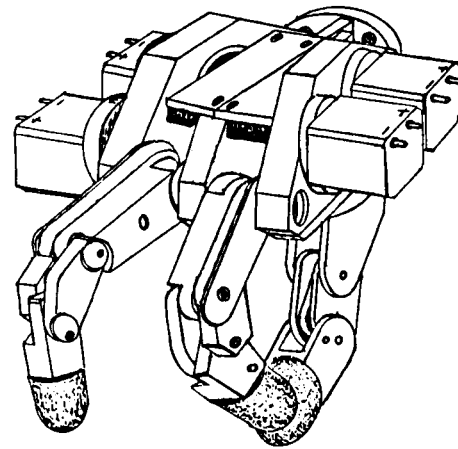


Figure 8. Three-Fingered Dexterous Slave Hand

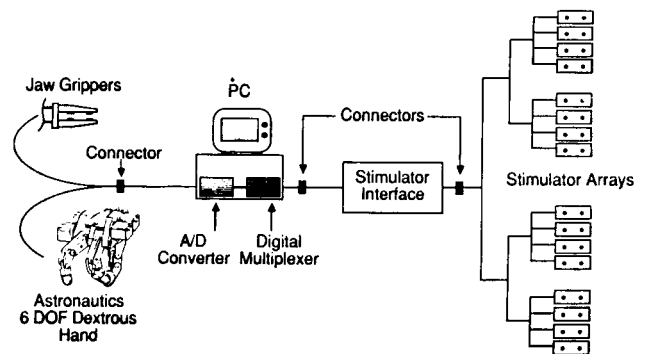


Figure 9. Tactile Feedback System

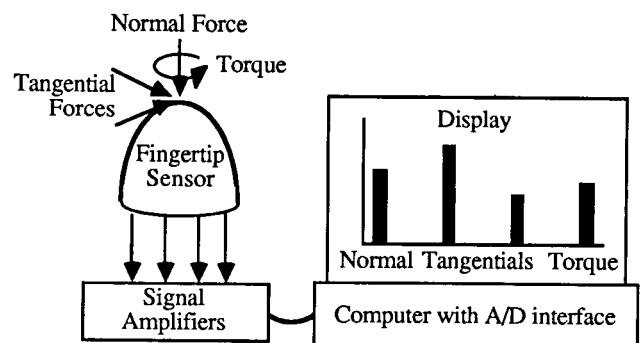


Figure 10. Fingertip sensor and signal processing system

HUMAN FACTORS EXPERIMENTS

Sustained operation of exoskeletal master controllers, particularly force-reflective systems, can be a fatiguing and uncomfortable experience if operational and feedback force levels are not properly adjusted for the individual operator. Though there is no guidance at this time for specifying these levels, previous exertion experiments have demonstrated that perceptions of force change with onset of localized muscle fatigue and discomfort [7,8]. One of WCSAR's objectives is to help develop a comfortable and fatigue-resistant master controller device which may be used in sustained manipulation and gripping activities by a wide ranging population.

WCSAR is now conducting a series of experiments, using apparatus such as that shown schematically in Figure 11. The objective is to determine the force-of-operation and end-effector force-feedback levels that can be endured for sustained periods in teleoperated systems (e.g. 2 hours) without encountering material signs and symptoms of fatigue and discomfort, and without altering the operator's perception of force produced or experienced at the end-effector. These experiments will help to produce performance response models which will account for differences in task duration, manipulation duty cycle (i.e. percent of time exerting against the manipulator), criticality of task force perception requirements, and an individual's pinch strength and psychometric sensitivity limitations.

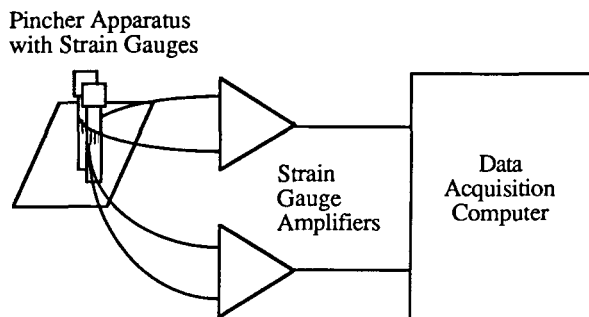


Figure 11. Apparatus for force-of-operation experiments

CONCLUSION

The work underway at the Wisconsin Center for Space Automation and Robotics (WCSAR) emphasizes the development of modular and add-on technologies that enhance dexterity, sensory perception, and performance in telerobotic systems. The component technologies described in this paper, and other related developments, are now being verified in the experimental telerobotic systems in the test bed. This will lead to their refinement and eventual availability for use in

telerobotic servicing systems being developed for space. We also expect significant terrestrial application spin-offs. Component developments being integrated into the WCSAR test bed will lead to enhanced capabilities at the system level. This will allow sophisticated human/sensor/actuator systems to be evaluated, with the results leading to practical systems with high levels of productivity and low levels of operator fatigue.

ACKNOWLEDGEMENTS

This work was supported by the Wisconsin Center for Space Automation and Robotics in part by NASA under grant NAG-W975 and by a consortium of industrial sponsors.

REFERENCES

- [1] Flight Telerobotic Servicer (FTS), Strawman Concept Engineering Report, Goddard Space Flight Center, Maryland, SS-GSFC-0031, March 1987.
- [2] French, R., and Boyce, B., "Satellite Servicing by Teleoperators," ASME Journal of Engineering for Industry, Vol. 107, pp. 49-54, Feb., 1985.
- [3] Mason, M., and Salisbury, J., Robot Hands and the Mechanics of Manipulation, The MIT Press, Cambridge, MA, 1985.
- [4] Jacobsen, S. et. al., "Design of the UTAH/M.I.T. Dexterous Hand," IEEE Intl. Conf. on Robotics and Auto., San Francisco, CA, April, 1986, pp. 1520-1532.
- [5] Lorenz, R., Meyer, M. and Van De Riet, D., "A Novel, Compliant, Four Degree-of-Freedom, Robotic Fingertip Sensor," to be presented at the IEEE-IAS Annual Meeting, Pittsburgh, PA, Oct. 3-7, 1988.
- [6] Badreldin, A., and Seireg, A., "Optimizing the Handling of Objects with Multi-Fingered Grippers," Computers in Mechanical Engineering, March/April, 1988, pp. 47-57.
- [7] Hightower, J., Smith, D., and Wiker, S., "Developments of Remote Presence Technology for Teleoperated Systems," Proc. of the 14th Meeting of the United States-Japan Natural Resources Comm. of the Marine Facilities Panel, Bethesda, MD, Sept. 19-20, 1988.
- [8] Sharum, J. and Chafin, D., "Force Judgement Decrements Following Muscle Exertions," Technical Report of the Human Performance Research Group, Dept. of Industrial Engineering, University of Michigan-Ann Arbor, 1970.

ORIGINAL PAGE IS
OF POOR QUALITY

THE USE OF THE ARTICULATED TOTAL BODY MODEL
AS A ROBOT DYNAMICS SIMULATION TOOL

Louise A. Obergefell
Xavier J. R. Avula
Ints Kaleps

Armstrong Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433-6573

ABSTRACT

The Articulated Total Body (ATB) model is a computer simulation program which was originally developed for the study of aircrew member dynamics during ejection from high-speed aircraft. This model is totally three-dimensional and is based on the rigid body dynamics of coupled systems which use Euler's equations of motion with constraint relations of the type employed in the Lagrange method. In this paper the use of the ATB model as a robot dynamics simulation tool is discussed and various simulations are demonstrated. For this purpose the ATB model has been modified to allow for the application of torques at the joints as functions of state variables of the system. Specifically, the motion of a robotic arm with six revolute articulations with joint torques prescribed as functions of angular displacement and angular velocity are demonstrated. The simulation procedures developed in this work may serve as valuable tools for analyzing robotic mechanisms, dynamic effects, joint load transmissions, feed-back control algorithms employed in the actuator control and end-effector trajectories.

INTRODUCTION

Work in the aerospace environment presents special problems which can be handled remotely by the use of automation techniques and robots. During aerospace operations, robot arms and hands can be controlled by a distant operator through exoskeletal devices to perform tasks such as repairing failed equipment, rescuing astronauts and handling hazardous materials. These tasks require extreme

manipulatability and dexterity. The development of the technology necessary to achieve the level of fine task performance required in aerospace operations involves an understanding of the three-dimensional kinematics and dynamics of robotic systems and of the control techniques for accurately manipulating these devices. At the Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, the Articulated Total Body (ATB) model has been successfully used in the investigation of manikin and human body dynamics. In view of the model's dynamic simulation capability and the similarities between robotic arms and the human arm, an attempt has been made in this study to add an active driving feature to the ATB model's passive response capabilities in order to use it as a dynamics and feedback control simulation tool.

DESCRIPTION OF THE ATB MODEL

The ATB model was originally developed as the Crash Victim Simulator (CVS) model for the National Highway Traffic Safety Administration (NHTSA) by Calspan Corporation in the early 1970's to predictively simulate occupant motion during automobile crashes (Ref. 1). It was subsequently modified to address Air Force requirements and renamed the ATB model (Refs. 2-5). It has been used extensively to study human and manikin body dynamics in aircraft ejections, automobile crashes and rollovers, and other mechanical force environments (Refs. 6-8).

The ATB model is based on rigid body dynamics, allowing a system to be described as a set of rigid segments, coupled at joints which allow the application of torques as functions of joint orientations and rate of change of orientations. A typical initial body

* Visiting Scientist on intergovernmental personnel assignment from the University of Missouri-Rolla.

configuration for a human or manikin simulation is shown in Figure 1. External forces are applied to the segments through interaction with other segments, contact planes used to describe the seat, floor, control panel, etc., belt restraint systems, pressure fields such as those due to wind forces, and gravity. Each segment has a surface approximated by an ellipsoid which is used to define a contact surface, application points for external forces and a reference for calculation of the contact forces. Motion constraints can also be placed on or between the segments.

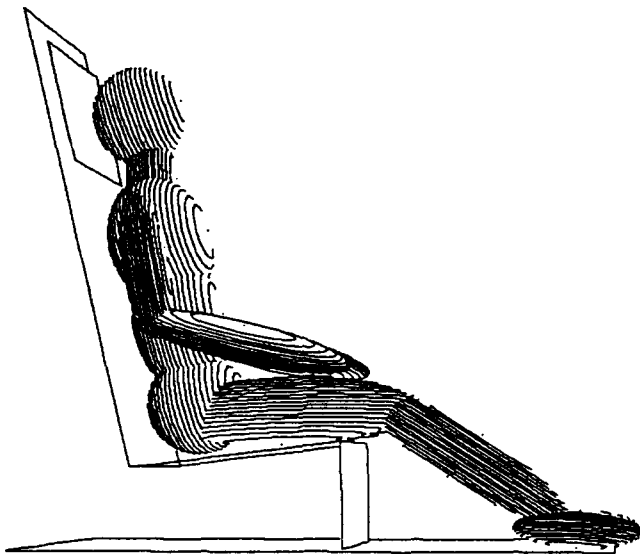


FIGURE 1. INITIAL BODY CONFIGURATION
FOR HUMAN OR MANIKIN ATB SIMULATION

Many complex dynamic systems that can be described in terms of multiple rigid bodies can be modeled with the ATB model because of its generality and flexibility. An input data set consisting of the geometrical, inertial and material properties of the segments; the joint characteristics; definition of the environment, such as contact planes, belts, wind forces and gravity; and time histories of known motions defines a specific simulation for the model.

The ATB model provides a wide variety of options for output, including the time history data for the motion of all segments, transferred joint forces and torques, and external interactive forces. Also the associated VIEW graphics program provides three-dimensional projected images of the system as shown in Figure 1 for the human body (Ref. 9).

JOINT ACTUATORS

The above described features make the ATB model an ideal tool for modeling the dynamics of robotic systems. However, in the ATB model, which was originally designed to predict passive response, the system of rigid bodies reacted to external forces caused by the prescribed environment. To simulate robotic systems, an active driving capability had to be added to the model.

Robotic systems have actuators such as motors driving each joint articulation. These actuators typically apply a torque to the joint that drives the joint to a specific position or through a trajectory. The torques are adjusted by the feedback algorithms of the system. The active driving components of robotic systems are such actuators. Therefore the capability to model actuator response was added to the ATB model as the active element.

The most common state variables used in feedback control are the joint position and velocity. The model uses the positions and velocities of the system of segments to calculate all the forces and torques on each segment at each integration time step. These forces and torques include contact forces between segments and between segments and other surfaces or belts, aerodynamic forces, gravity and joint resistive torques. Since the actuators need this same information for the feedback algorithms, the actuator torque calculation was added to this part of the program. The program has been set up to feed back joint angle and velocity, enabling the use of position, derivative and integral control. At each time step in the program all the state variables are known and can be used as feedback variables for the actuators. Therefore variables such as linear positions or forces may also be used for feedback.

The actuator feedback calculation is contained in a subroutine that the user can modify to model the feedback algorithm required. Without modifying this subroutine, there is still considerable flexibility in the feedback provided by simply by changing the feedback parameters in the program input program input.

ROBOT SIMULATION

To test and demonstrate the use of the ATB model as a robotic simulator, an example robot with six articulations has been simulated. The input requirements for this simulation are representative of those for any robotic simulator including

ORIGINAL PAGE IS OF POOR QUALITY

spatial geometry, inertial properties and joint position control information. The results of the simulations made of this robot demonstrate the ability of the ATB model to predict typical control system responses while taking into account the effects of inertial properties and gravity on system response.

Simulation Specification

To simulate any system the ATB model requires an input file describing that system and the surfaces that it may contact. The data describing the system consists of the mass, moments of inertia and geometry of each rigid link, the location and rotation axis orientation of each articulating joint and the characteristics of each actuator. The robot simulated is based on an American Cimflex MR6500 Merlin robot and the model's depiction of it is shown in Figure 2 with its six joints labelled. Mass and moment of inertia data were estimated from the limited mass data and geometric data available on the robot. For this simulation the planes and ellipsoids associated with the segments are used only for graphical display. If contact by a robot segment with another object was to be simulated the geometrical elements could be used to determine whether contact was occurring, the contact point on the segment and the contact forces. All of the segment and joint data are prescribed in each of the segments' local coordinate systems,

located at the segment center of mass. The joint locations and rotations axes orientations were measured and prescribed with respect to these local coordinate systems. The robot is shown in its home position and its articulations are defined as: waist yaw at joint 1, shoulder pitch at joint 2, elbow pitch at joint 3, forearm roll at joint 4, wrist pitch at joint 5 and wrist roll at joint 6.

Each joint was assigned an actuator, which applied torques as functions of the joint position variables, about the respective joint axes. The form of the torque feedback algorithm for each actuator used in the initial simulations is:

$$T = f_2(\theta - \theta_0) - f_3(\dot{\theta}) \quad (1)$$

$$\text{Where: } \theta_0 = f_1(t),$$

T is the joint torque applied by the actuator,

θ is the joint angle,

θ_0 is the joint target angle,

$\dot{\theta}$ is the joint angular velocity,

t is time, and

f_i are input functions.

The input functions can have a variety of forms including a constant, polynomial, tabular or combination. Simple functions were chosen for these simulations to test the program. The functions used were:

$$f_1(x) = a$$

$$f_2(x) = bx$$

$$f_3(x) = cx$$

The constants a , b and c used for each joint were varied to demonstrate different system responses.

Results

The robot motion for a simulation in which all the joints were driven to different angles is shown in Figure 3. The graphics program allows the simulated system to be displayed at any time step and from any viewing angle. The simulation also provides time history data on the segment positions and orientations, the joint orientations and torques, and the actuator torques. Figure 4 contains plots of all of the joint

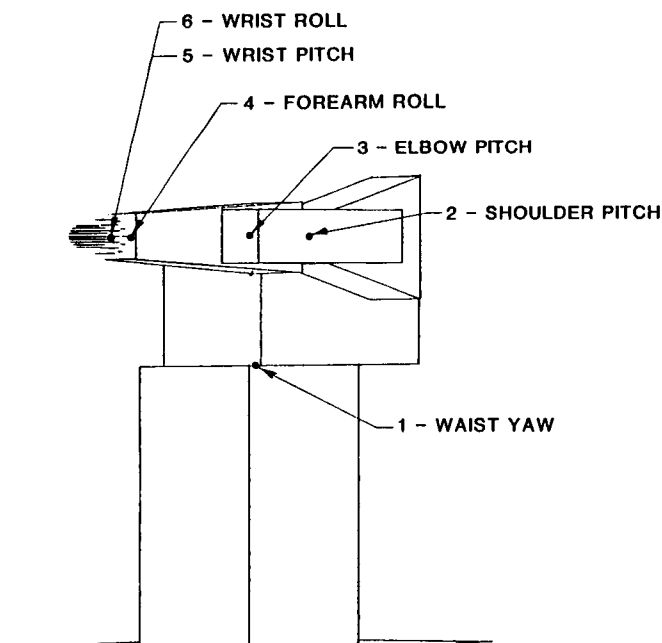


FIGURE 2. ROBOT ARM WITH SIX JOINTS

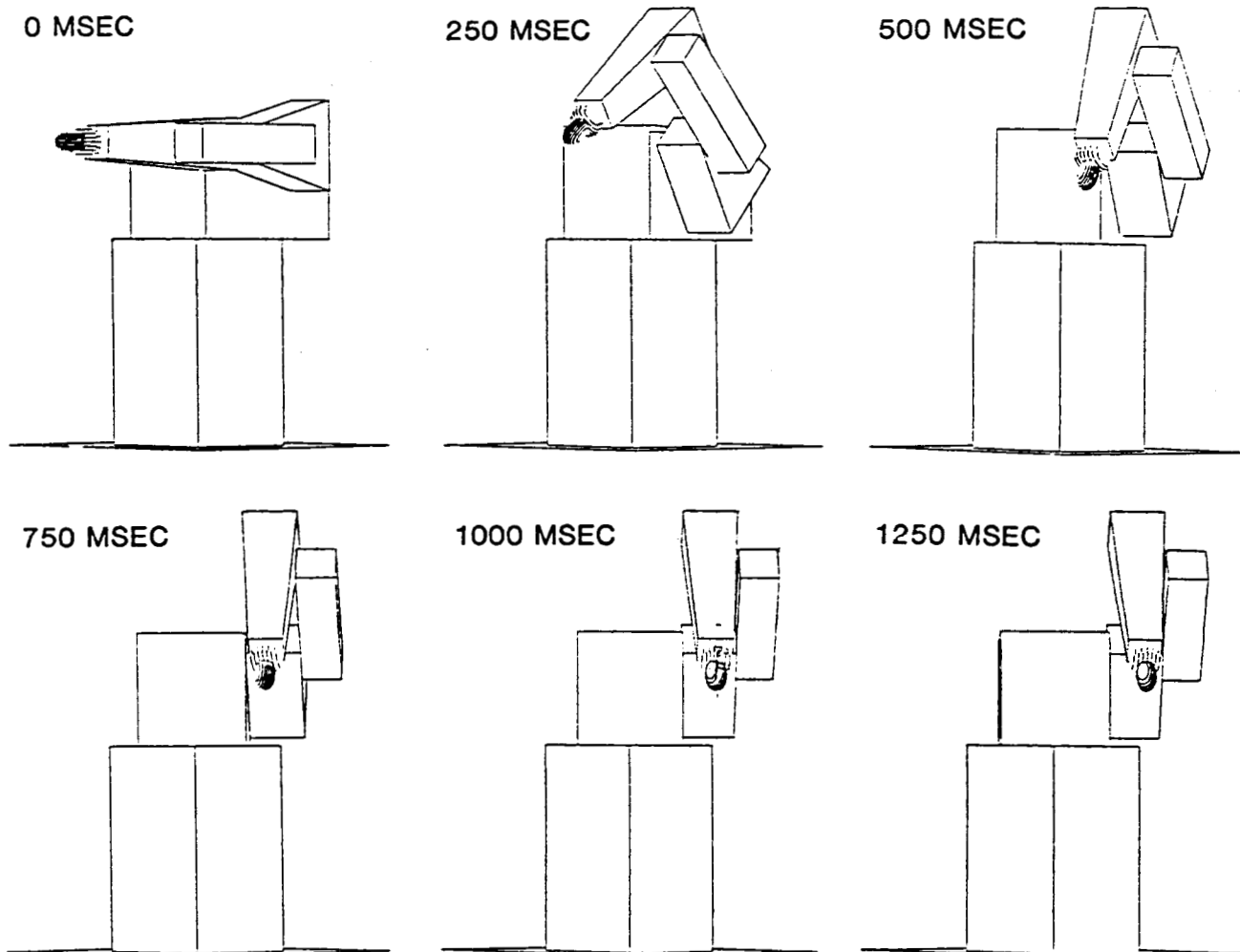


FIGURE 3. SIMULATED ROBOT MOTION

angles for the above simulation. These plots demonstrate several important characteristics of a dynamic simulation. The wrist pitch target angle was zero degrees, but the wrist does pitch slightly during the first 400 msec. due to the motion of the other joints. The shoulder pitch levels off at an angle slightly less than its 45 degrees target angle and the elbow pitch levels off at an angle slightly more than its 90 degrees target angle due to the torque required at each of these joints to compensate for the weight of the arm. It is also likely that the shape of the forearm roll plot is affected by the wrist roll.

Figure 5 contains plots from four simulations in which the wrist roll actuator was driven to 90 degrees and all

the other actuators were driven to zero. The feedback parameters for the wrist roll actuator were varied to obtain the different wrist roll responses seen in the plots. The differences in the other joints' motions again demonstrate the inertial effects of the system. The forearm roll is especially affected by the large motions of the wrist.

DISCUSSION

In this study, we have demonstrated that the ATB model, with the active driving capability of the actuator modifications, can be used as a robotic dynamics simulation tool. It is intrinsic to the ATB program to account for the dynamic characteristics (or inertial effects) of the arm, as exhibited by the time histories of the various joint motion in

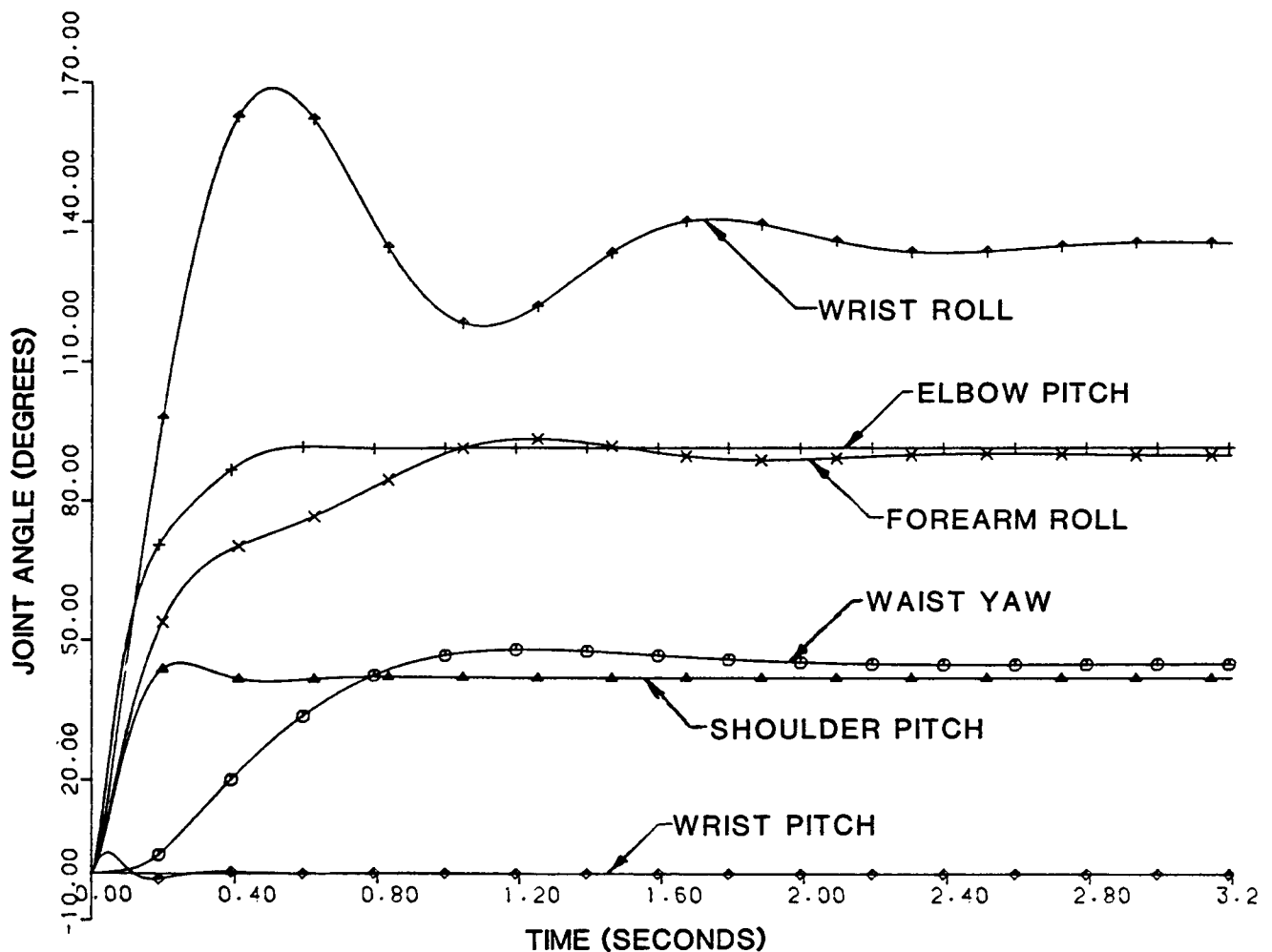


FIGURE 4. JOINT ANGLE RESPONSES

the robotic arm simulation (Figures 4 and 5). Although the segment yaw, pitch and roll angles are kinematic quantities, a pure kinematic simulation would not predict the responses demonstrated here due to its neglect of the inertia properties of the system. Bringing out the dynamic characteristics of the system under simulation has been proved to be one of several strengths of the ATB model. With its capability to incorporate a variety of environmental forces and torques and its flexibility to model different system structures, the model has been established to be a versatile tool for further development of robotic simulation methods.

Future work with the ATB model, could allow investigations of integral control, control algorithms which couple the motions of several joints, force control, and adaptive control. Because the model calculates all the state variables needed for each of these control methods at each time step, their dependence can easily be incorporated into the feedback subroutine developed in this study.

The next logical step in this work is a validation of the model predictions. This can be accomplished by exercising a robot with the same structure, inertial properties and feedback algorithms and comparing its responses with those of the model.

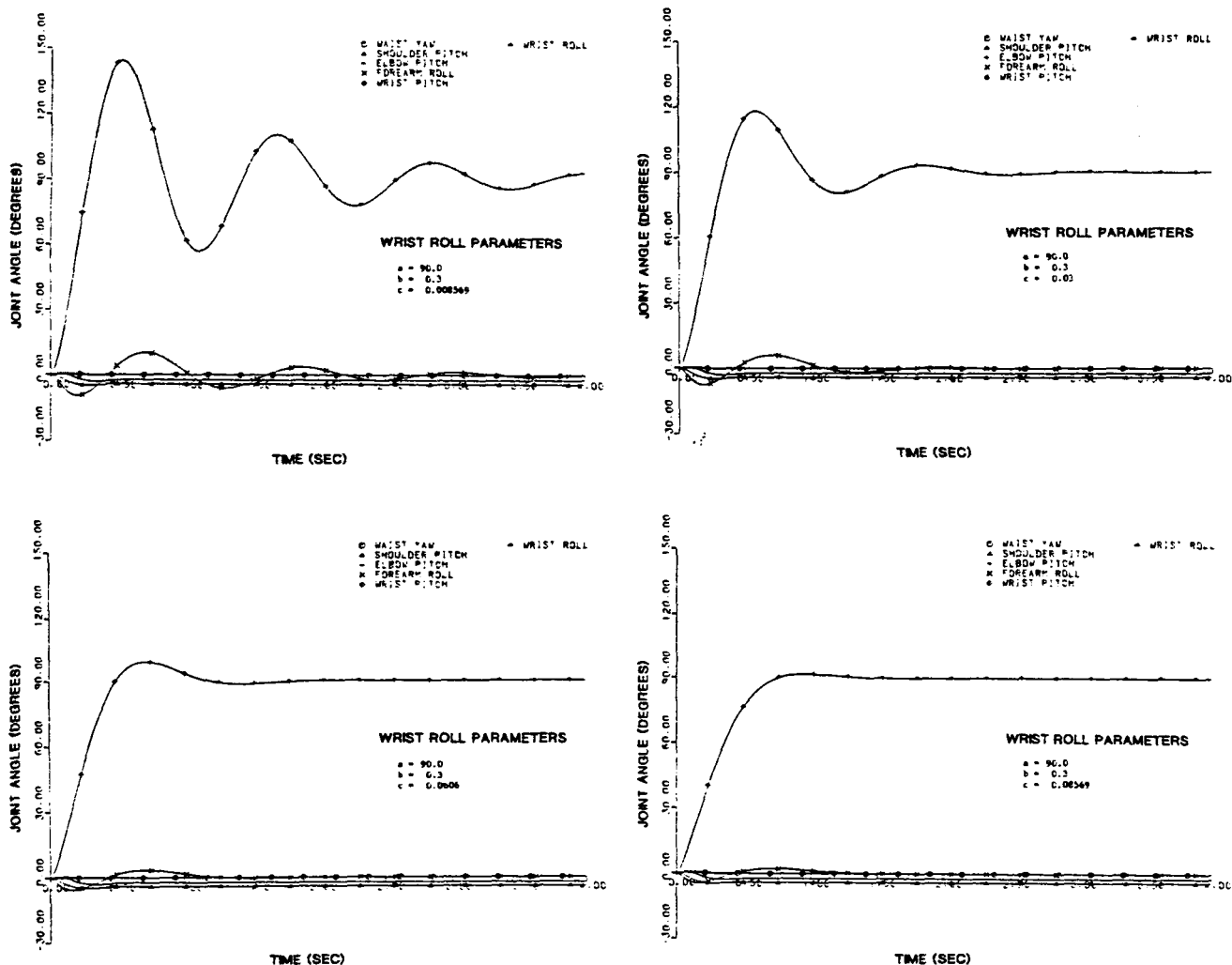


FIGURE 5. JOINT ANGLE RESPONSES

VARYING WRIST ROLL ACTUATOR FEEDBACK PARAMETERS

REFERENCES

1. Fleck, J.T., Butler, F.E., and DeLays, N.J., "Validation of the Crash Victim Simulator," Report Nos. DOT-HS-806-279 thru 282, Vols 1-4, 1982.
2. Fleck, J. T. and Butler, F. E., "Development of an Improved Computer Model of the Human Body and Extremity Dynamics," Report No. AMRL-TR-75-14, April 1975.
3. Butler, F. E. and Fleck, J. T., "Advanced Restraint System Modeling," Report No. AFAMRL-TR-80-14, May 1980.
4. Obergefell, L.A., Kaleps, I., Gardner, T.R. and Fleck, J.T., "Articulated Total Body Model Enhancements, Volume 3: Programmer's Guide," Report No. AAMRL-TR-88-007, Feb. 1988.
5. Obergefell, L.A., Fleck, J.T., Kaleps, I. and Gardner, T.R., "Articulated Total Body Model Enhancements, Volume 1: Modifications," Report No. AAMRL-TR-88-009, Jan. 1988.
6. Kaleps, I. and Marcus, J. H., "Predictions of Child Motion During Panic Braking and Impact," Proceedings 26th Stapp Car Crash Conference, SAE Paper No. 821166, Oct. 1982.

7. Obergefell, L. A., Kaleps, I. and Johnson, A.K., "Prediction of an Occupant's Motion During Rollover Crashes," Proceedings 30th Stapp Car Crash Conference, SAE Paper No. 861876, Oct. 1986.
8. Obergefell, L. A. and Kaleps, I., "Simulation of Body Motion During Aircraft Ejection," Mathematical Modelling in Science and Technology, Proceedings 6th International Conference, Aug. 1987.
9. Leetch, B. D. and Bowman, W. L., "Articulated Total Body (ATB) VIEW Package Software Report," Report No. AFAMRL-TR-81-111, Vols 1 and 2, Oct. 1981.

TELEPRESENCE AND TELEROBOTICS

John Garin,
Joseph Matteo,
and
Von Ayre Jennings, Ph.D.

Martin Marietta Aero & Naval Systems
103 Chesapeake Park Plaza
Baltimore, MD 21220

ABSTRACT

Martin Marietta is developing the capability for a single operator to simultaneously control complex remote multi degree of freedom robotic arms and associated dextrous end effectors. An optimal solution within the realm of current technology, can be achieved by recognizing that (1) machines/computer systems are more effective than humans when the task is routine and specified, and (2) humans process complex data sets and deal with the unpredictable better than machines. These observations lead naturally to a philosophy in which the human's role becomes a higher level function associated with planning, teaching, initiating, monitoring, and intervening when the machine gets into trouble, while the machine performs the codifiable tasks with deliberate efficiency.

This concept forms the basis for the integration of man and telerobotics, i.e., robotics with the operator in the control loop. The concept of integration of the human in the loop and maximizing the feed-forward and feed-back data flow is referred to as telepresence.

Telepresence at Martin Marietta consists of an exoskeleton master commanding an anthropomorphic slave robot. The slave will serve as a human surrogate, replacing man in hostile environments. The exoskeleton master controller will provide a feeling of transparency (operator believes he is in the robot's surroundings) through advanced controls. This approach will address integration of the human into the control loop.

INTRODUCTION

The new, high technology battlefield has become an extremely lethal environment where survival will depend on human surrogate devices. These hazardous environments are such that with the human protected by armor and/or environmental suits (i.e., space suit, diving equipment), performance is greatly degraded and the human life is under unacceptable risk. In some cases human access may be impossible.

Hazardous environments such as Nuclear-Biological-Chemical (NBC) environment, subsea or outer space, will require robotic systems that can perform human-like tasks. The performance of these tasks will require robotic arms that have redundant kinematics, bandwidth, and weight-to-power ratios of the human.

Modern robotic arms that resemble humans have seven or more degrees of freedom (DOF) to allow arbitrary positioning and orientation of the end effector. Manipulators with more than six DOF can have control problems as the manipulator configuration often becomes degenerate (two or more DOF produce the same motion of the end effector). We are addressing robotic arm requirements of 7 DOF. This type of kinematic arrangement is required to perform human-like tasks and provide telepresence functions.

Redundant kinematics significantly affects telerobotics control, and requires special man-machine interfaces. To avoid degeneracies, a special controller must be used, one type of which (an exoskeleton controller) Martin Marietta is now developing.

There are basically three (3) types of kinematic stances used in telerobotics. The nuclear industry typically uses the "elbow up" stance. The upper arm extends horizontally outward, lower arm dropping vertically downward, and the end effector extending horizontally.

The Japanese have implemented a design that has the "elbow sideways" or the actuators causing motion in a horizontal plane. This kinematic arrangement is referred to as a Selective Compliance Articulated Robotic Arm (SCARA). It is very useful for industrial applications because motion is independent of gravity, but has limited use in telerobotics.

The third configuration is the anthropomorphic stance which resembles that of the human arm. The upper arm drops vertically, and the lower arm and end effector axes are aligned, extending forward horizontally. Martin Marietta will be concentrating on the anthropomorphic stance of the human as their model. The choice affects control issues greatly and is based on the rationale in the following paragraph (Figure 1).

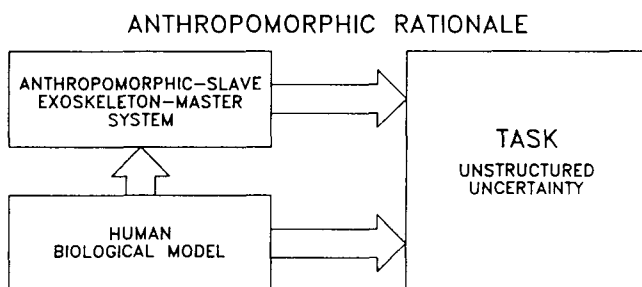


Figure 1. Teleoperator System Modelled After Human Operators will be More Intuitive, Easier to Use, Reliably Controlled

A telerobotic system that is modelled after the human operator will be more intuitive, and thus easier to use (less cognitive input) and more reliably controlled (less fatigue to the operator). This reduces learning time, and creates a more natural control environment.

TELEPRESENCE AND TELEROBOTICS

Martin Marietta Aero and Naval Systems started the development of telepresence systems in 1986.

The objective of the Aero & Naval Systems Exo-skeleton Master and Anthropomorphic Slave (EMAS) arms system is to provide dextrous manipulation. This system will utilize and develop the man-machine interface and be used in a tele-operational control mode to provide maximum telepresence effect: sensory information back to the operator.

The term telepresence refers to the information required by the operator to "feel" a sense of "presence" in the working environment. The issues dealing with telepresence relate to three major human factor issues:

- Perception
- Cognition
- Psychomotor Control

To complete any task, whether teleoperated or not, the task must first be understood (perception), a decision must be made about what action to take (cognition), and then that action must be carried out (psychomotor control). This same thought process is paralleled in the robot controller.

The ultimate goal of this system is to provide human-like manipulation capabilities. In some cases, physical capabilities will exceed those of humans.

The measure of these capabilities can be summed into dexterity which refers to the ability of the manipulator to perform tasks that require human-like manipulative capabilities. The robotic arms will provide that type of manipulation. The degree of dexterity will be characterized by comparing the time to perform this task by a human with his/her bare hands to that of the robotic arms. The measured parameters will be:

- rate of task completion
- accuracy/quality of task performance
- impact of system on remote environment
- impact on operator, i.e. workload/fatigue

The overall relationship between the major components of the exoskeleton master and anthropomorphic slave are shown in Figure 2.

ANTHROPOMORPHIC SLAVE

The anthropomorphic slave arm will approximate human performance and be scaled in both size and strength. Specifically, a single manipulator can handle an object up to 50 pounds in weight and

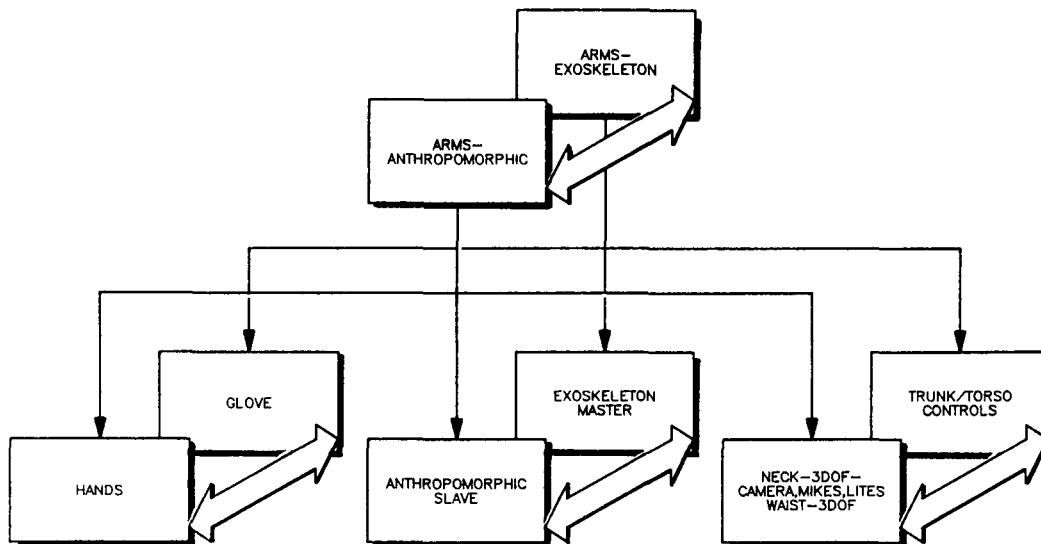


Figure 2. Telepresence System Exoskeleton Master and Anthropomorphic Slave

move it throughout most of a hemisphere with a radius of approximately 4 feet. The total weight of 1.5x scaled up slave arm should be no more than approximately 160 pounds. For portability, it can be quickly disassembled into modules weighing no more than 40 pounds each. The most demanding sequence of operations involving motion of a heavy (approaching capacity) object, grasping the object, and reorienting and moving the object across the manipulator range of motion to a new location can be accomplished in less than 5 seconds with great accuracy. At lesser load, the arm can reach speeds of 160 ft/sec² and 360 ft/sec² tip speed carrying 25 lb. and 0 lb. payloads, respectively. The single arm manipulator system is able to meet the performance specifications while mounted on a platform, a ROV submersible, or on the back of a light truck and while subjected to an adverse field environment.

EXOSKELETON MASTER

The exoskeleton master goals include being used by 5-95th percentile operators. This is achieved by providing adjustments in the major anthropometric ranges. These ranges of adjustments are still being determined.

The following information and sensory feedback shall be provided back to the operator (Figure 3).

Hand Controller (Master)

1. Force Reflection
2. Somesthetic Feedback
 - Tactile
 - Pressure (below skin surface, deep muscular pressure)
 - Temperature
3. Joint Angle Sensing
4. Haptic Display

Arm Controller (Exoskeleton Master)

1. Force Reflection
2. Joint Angle Sensing
3. Stiffness Sensing (Kinesthesia)
 - EMG
 - Muscle Sorno
 - Co-Contraction

SERVO CONTROL REQUIREMENTS

Innovation in servo-control system design has been under development. This architecture is as shown in Figure 4. Control of both force and position are required. The system must adapt to payload, be achievable on a flexible or moving platform, and driven by sensors of position and force. Control algorithms are arranged for rapid computation on distributed processing computers. The following (4) control modes are provided.

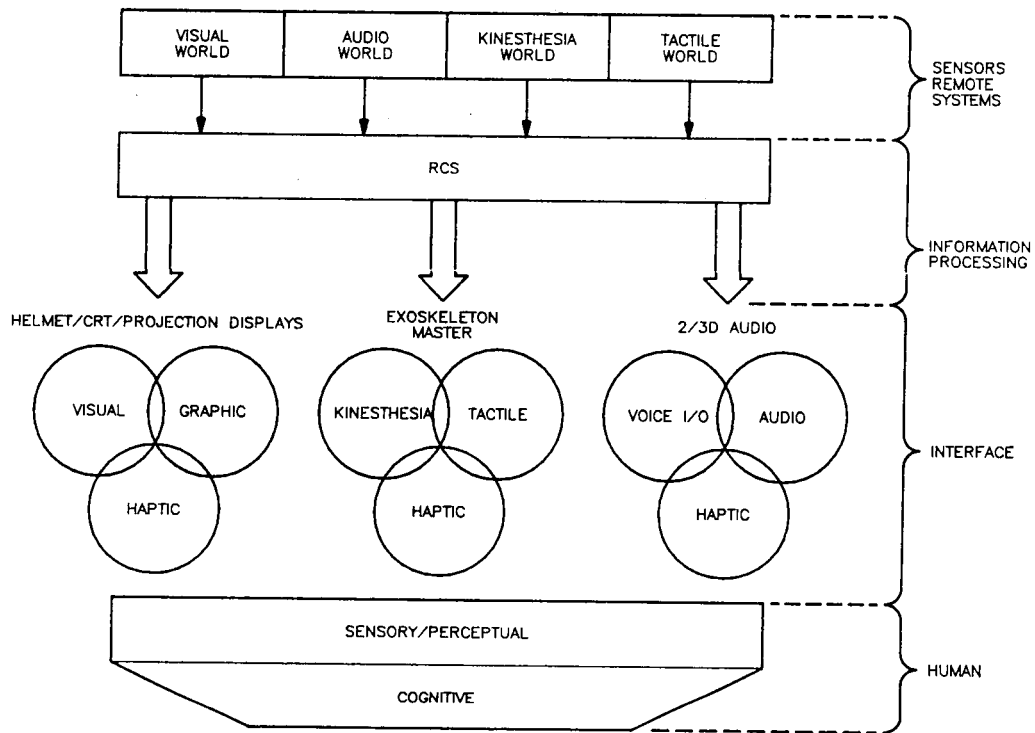


Figure 3. Telepresence Control Station Feed Back

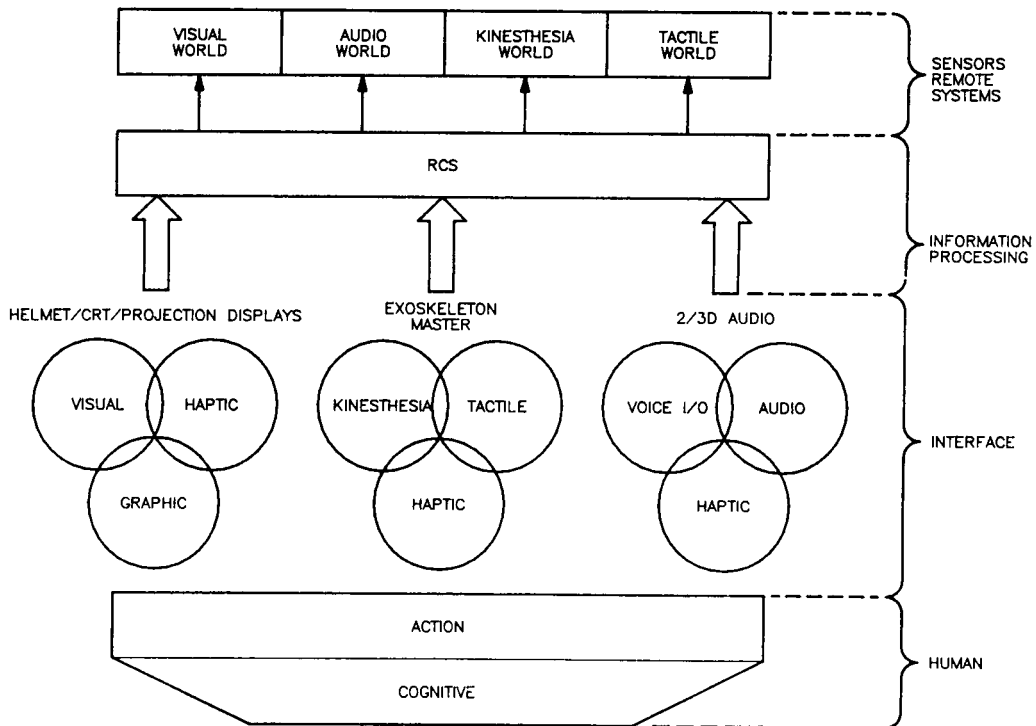


Figure 4. Telepresence Control Station Feed Forward

- **Computer Supervisory Control** – The slave arms shall be able to perform preprogrammed functions such as:
 - manipulator arm storage
 - tool extraction and storage
 - repetitive tasks that have been previously taught
 - collision avoidance (arm to arm)
 - end of arm camera tracking
- **Control Adaptive to Accuracy** – Speed of motion is faster when less accuracy of position or force control is required.
- **Cooperation Control** – Two arms are able to cooperate on the handling of a long and heavy object and on the application of opposing forces to one or two objects.
- **Collision Survival** – The arm will stop motion without damage to itself upon contacting an obstruction to the motion of any link of the arm, when the speed of motion is one quarter of the highest speed.

INTEGRATED WORK STATION

For telerobotics to be a cohesive system or provide telepresence, four key perceptual functions, visual, audio, kinesthesia and tactile world (Figures 3 and 4), must be integrated.

The visual feedback will be achieved through a video system with enough resolution to match operator acuity. The audio system will provide real-time auditory information with 3D cues. The kinesthesia information is provided through the exoskeletal controller in the form of forces and torques. The tactile world is addressed through touch sensors on the surfaces of the arms and fingers.

- a) Development of a comprehensive concept for an anthropomorphic manipulator system (Figure 5). The overall concept for this system consists of the slave manipulator, an exoskeleton master controller, telepresence sensory displays and a computer control system. The underlying approach to the concept is to integrate the human operator into the system in a way that takes advantage of the human's tremendous perceptual, cognitive and psychomotor capabilities. For example, the manipulator arm has 7 degrees of freedom that are identical to the arrangement of joints on the human arm. Similarly, forces and torques acting on the manipulator will be fed back to the human's arm in a way that will cause operators to feel as if they were doing the manipulator task themselves. In this way, the anthropomorphic approach will allow the operator to concentrate on the task to be done instead of what motors to turn on or off.

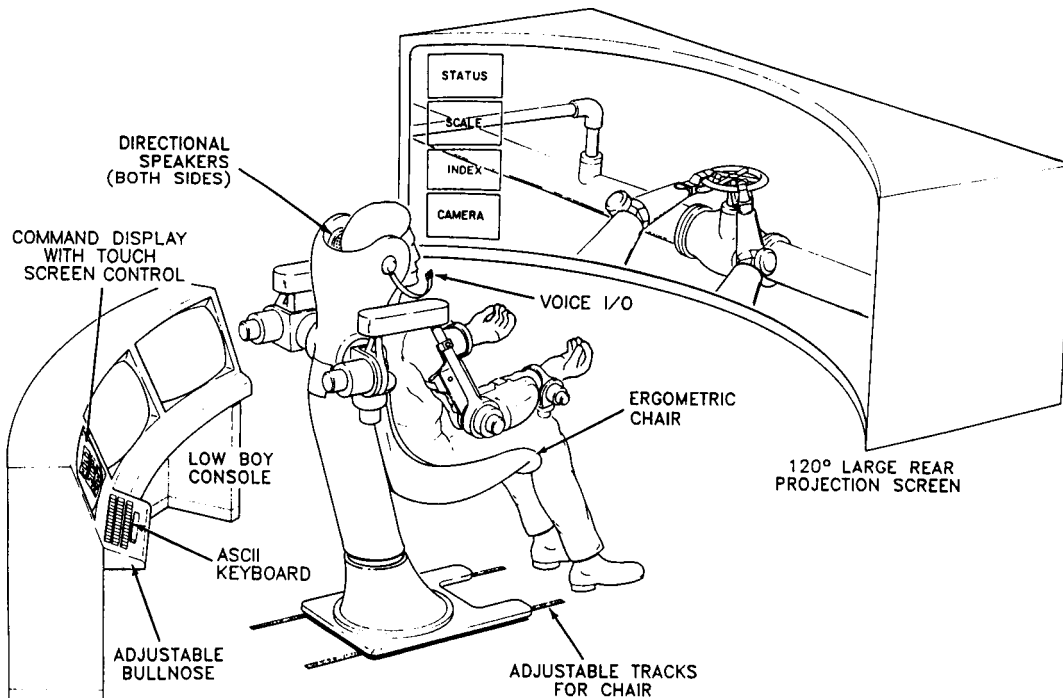


Figure 5. Telepresence Control Concept

- b) Definition of Functional Requirements for Anthropomorphic Performance. Several psychomotor and perceptual characteristics of the human operator were considered in defining the system functional requirements. For example, the movement characteristics of single joints in humans were evaluated to help define the limiting conditions for the corresponding manipulator joint (Figure 6). Other parameters examined included range of motion, accuracy, repeatability, static and dynamic torque for joint movements and a variety of psychophysical characteristics such as position and force sensitivity.
- c) Technology assessment and trade-off studies for actuators, sensors and power transmission. Using the functional requirements derived from

anthropomorphic considerations, various manipulator parameters were defined and analyzed. These include the joint ranges of motion, static and dynamic torques (Figure 7), and actuator optimization (Figure 8).

- d) Completion of Final Mechanical Design. The final design of the anthropomorphic manipulator, based on the design requirements described above, consists of a series of interchangeable modules that can be configured not only as an anthropomorphic design but in a variety of other configurations (Figure 9). This manipulator will serve as a unique test bed that represents a critical first step towards teleoperated control of highly maneuverable robot systems.

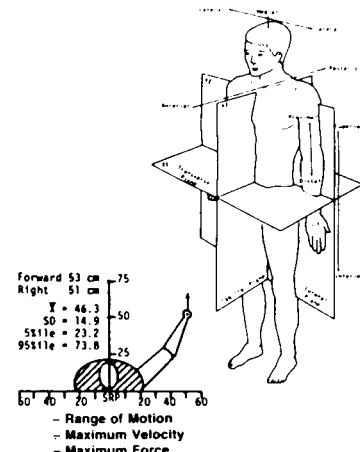
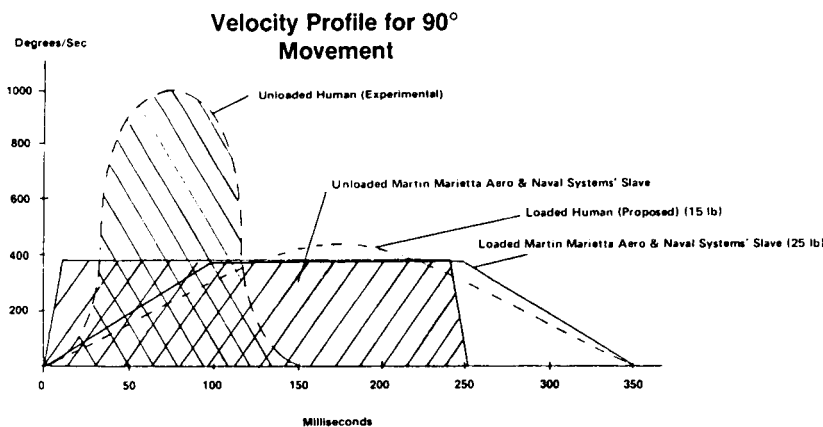
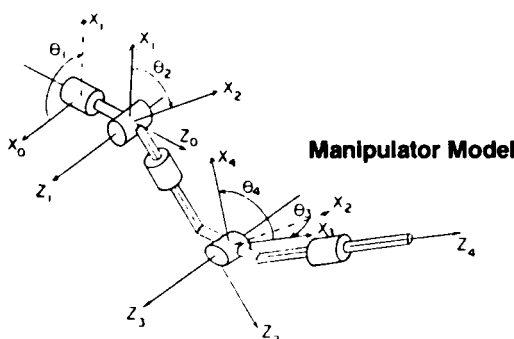


Figure 6. Human Performance Characteristics



RANGE OF MOTION		
θ_1	0° TO	180°
θ_2	-90° TO	+90°
θ_3	-90° TO	+90°
θ_4	0° TO	180°

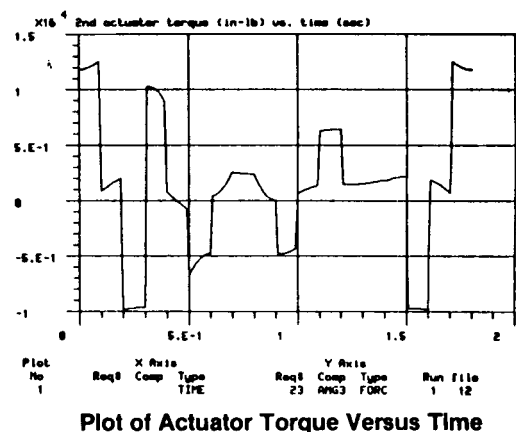


Figure 7. Analysis of Manipulator Dynamics

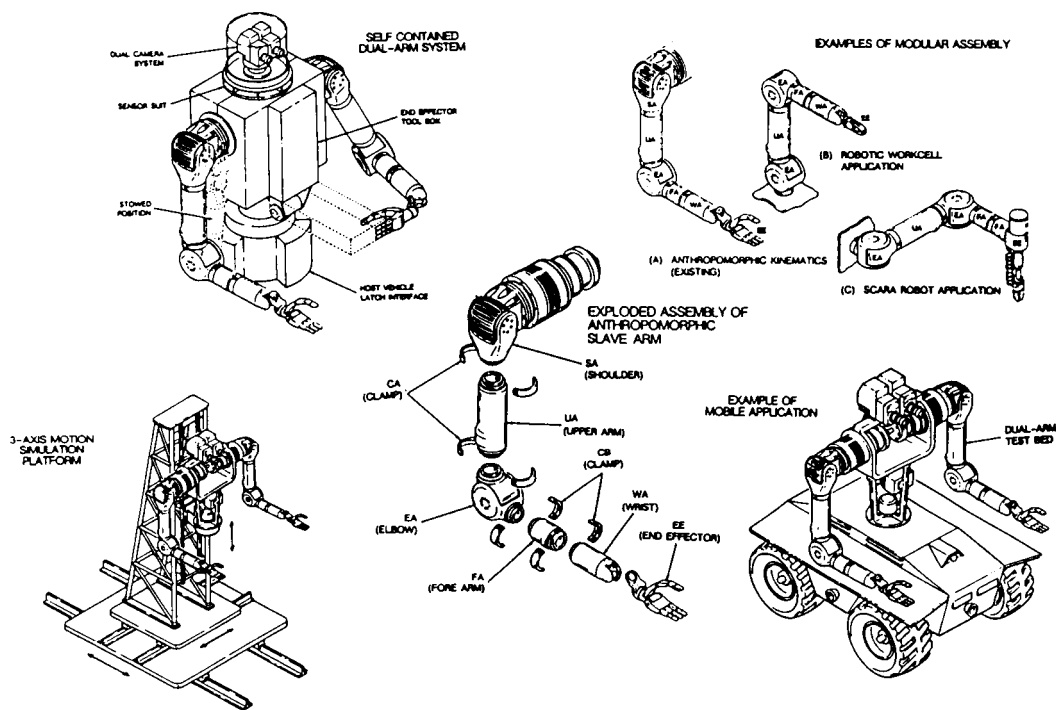
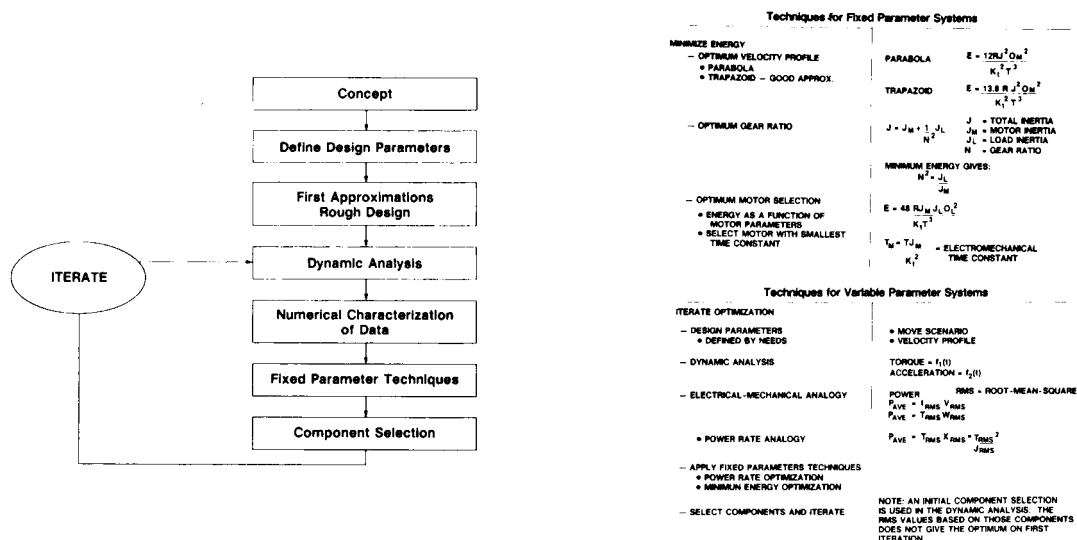


Figure 9. Anthropomorphic Manipulator

1) Conceptual design of the exoskeleton master controller is shown in Figure 5. The exoskeleton controller will be a unique method for controlling the anthropomorphic manipulator by using movements of a human operator's arm. This form of teleoperator control will also allow the forces and torques acting on the manipulator

to be directly applied to the operator's arm, resulting in an intuitive and easy to understand presentation of complex sensory data.

2) A test bed for teleoperator control of the anthropomorphic manipulator is being developed initially using a single degree of freedom

master-slave arrangement (Figure 10). This will consist of the elbow joint from the manipulator with its associated sensors and servo control and the elbow joint of the exoskeleton master device (described in the previous section). This test bed will be used to develop force reflection algorithms and algorithms for position or force control of the manipulator. These algorithms will allow a unique and powerful form of robot control to be tested, where human operators command robot movements by performing the movement themselves. This will provide initial information of kinesthesia.

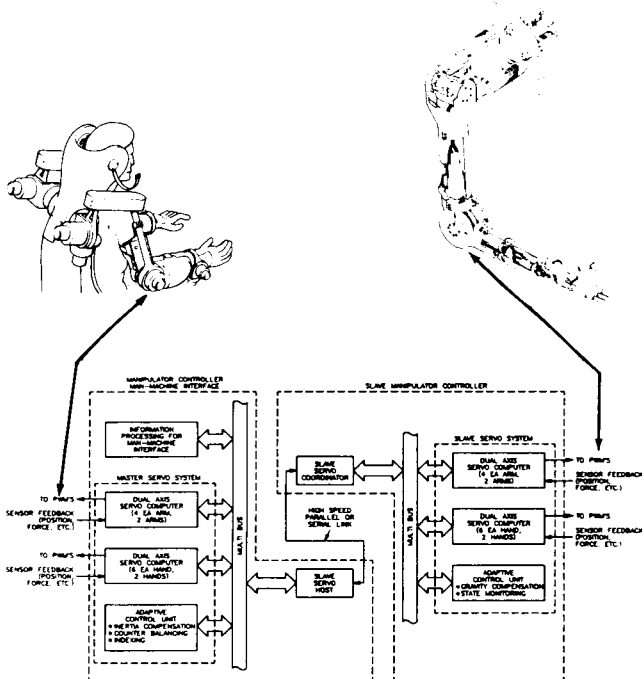


Figure 10. Telepresence Anthropomorphic Master/Slave Test Bed

The tactile information will be provided through a glove like device to feed information back to the operator's hands (see Figure 5).

We have proceeded to develop the manipulator system and the control station as described below.

Significant progress was made in two areas:

1) Design of anthropomorphic manipulator system:

The design of a unique 7 degree-of-freedom manipulator was completed. This manipulator, when constructed this year, will be one of the most advanced robot arms available in terms of speed, dexterity, accuracy and load-to-weight ratio. Progress was made in several key areas that contributed to the overall design.

2) Control approach

The control development has also progressed. The Advanced Servo Computer (ASC),

Figure 11, developed in 1986 was enhanced to address the needs of teleoperation.

The ASC board is an extremely powerful data acquisition and processing system that will allow a variety of high-speed and sophisticated low-level servo control algorithms to be implemented.

The ASC board will be used as shown in Figure 11 as part of the low level control for the master/slave system. The RCS (Real-time Control) approach is being implemented as shown in Figure 12. The functionality is also shown in Figures 13 and 14.

The basic concept is to utilize a powerful servo board which can handle sensory inputs and communications. As the cycle time to complete these tasks increases, the system stability increases and allows the use of more simple models.

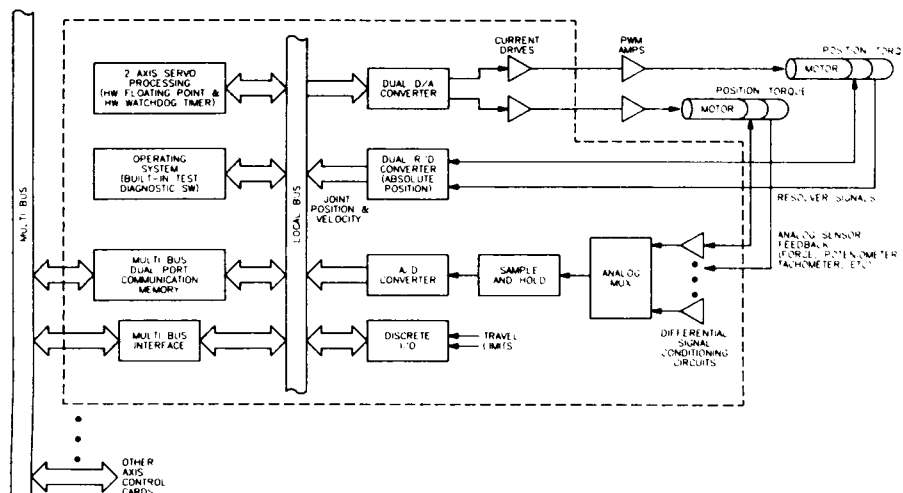


Figure 11. Dual Axis Servo Control Card

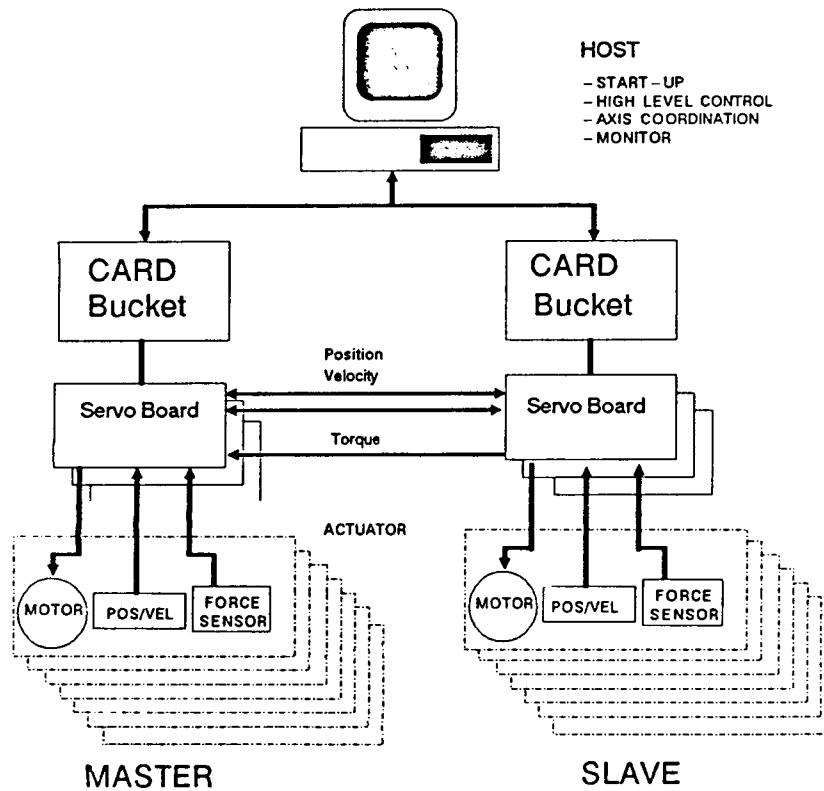


Figure 12. DOF Control Architecture

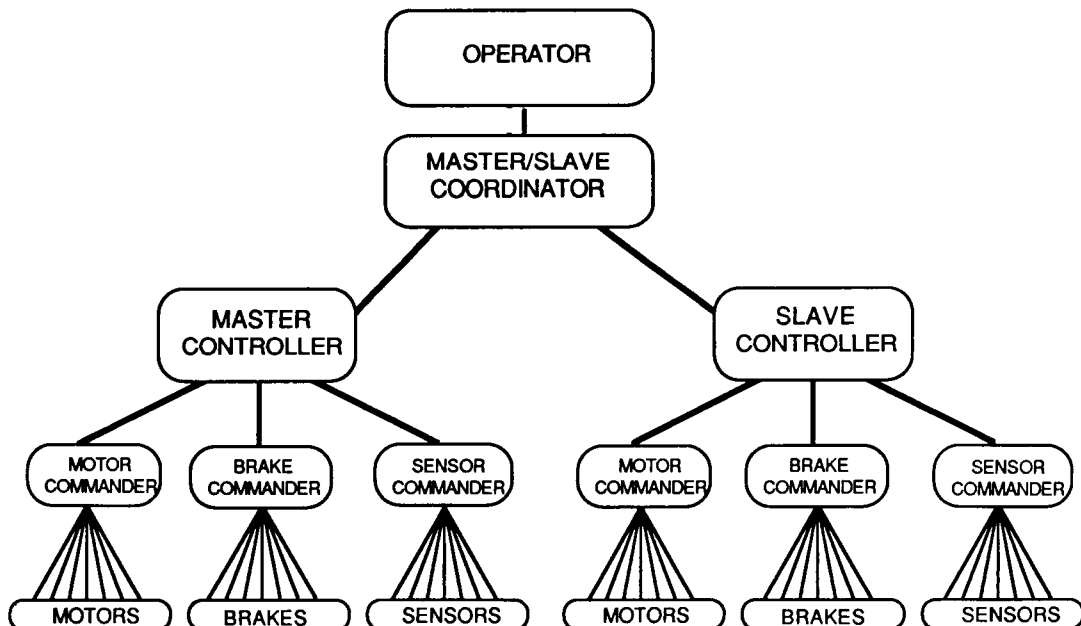


Figure 13. RCS Task Decomposition - Master/Slave

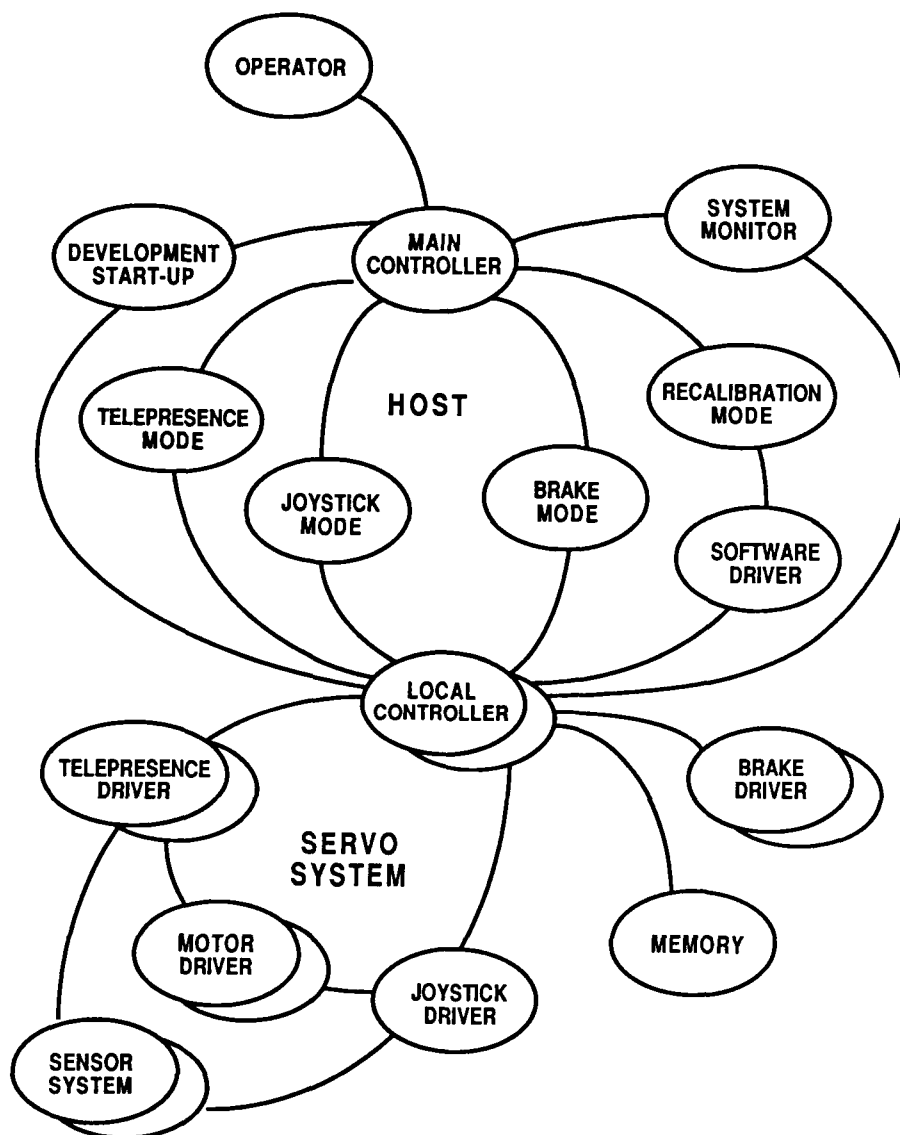


Figure 14. Data Flow

CONCLUSION

In conclusion, Martin Marietta is developing a significant capability to develop and test new concepts for man-machine interface and telerobotic systems. Through these test beds, the 1 DOF and 7 DOF systems, we will be developing the solutions for telepresence and optimizing methods of control and performance of teleoperations. The intention is to provide systems that will truly be functional human surrogates in hazardous environments.

The mechanical design uses state-of-the-art actuation components in an extremely compact design (patent pending). Special techniques were

developed to optimize system components, and often custom-made parts are used.

The control system is designed to allow investigation into many types of tele-operation control. The system is based around a powerful and flexible servo controller uniquely designed for teleoperation. The online system's approach addresses the basic premise of teleoperation (man is best for the unstructured thinking, the robot is best for repetition) by keeping the algorithms simple, improving the sensory feed back to the human, and allowing him to make natural high level and high speed inputs to the controller.

A Novel Manipulator Technology For Space Applications

Donald Schmitz, Pradeep Khosla, and Takeo Kanade

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Modular manipulator designs have long been considered for use as research tools, and as the basis for easily modified industrial manipulators. In these manipulators the links and joints are discrete and modular components that can be assembled into a desired manipulator configuration. As hardware advances have made actual modular manipulators practical, various capabilities of such manipulators have gained interest. Particularly desirable is the ability to rapidly *reconfigure* such a manipulator, in order to custom tailor it to specific tasks. This reconfiguration greatly enhances the capability of a given amount of manipulator hardware. This paper discusses the development of a prototype modular manipulator and the implementation of a configuration independent manipulator kinematics algorithm used for path planning in the prototype.

1. Introduction

The major advantage of robotic manipulators over task-specific hardware for automation is their flexibility. In theory, a robot's task can be changed simply by loading a new program into its controller. However, in practice this is rarely the case. Each robot has a specific configuration that supports a limited range of capabilities, appropriate only to the applications for which it was designed. The major factors that define the configurations are the link lengths, joint actuators, and geometry of joint-link connections. For example, horizontal SCARA-configuration manipulators, connected with relatively short links, are suitable for delicate table-top assembly operations requiring accuracy and selective stiffness, but they are not usable for tasks that require a vertically large workspace. On the other hand, medium-sized, vertical Puma-configuration manipulators with a relatively long reach in all directions, are suitable for painting, welding and parts handling. Using manipulators with different configurations for each task is possible when the task requirements are known beforehand. However, in less predictable situations, such as an outdoor construction site, inside a nuclear facility or aboard a space station, a manipulator system would need a wide range of capabilities, probably beyond the limitations of a single fixed-configuration manipulator.

We have proposed a manipulator system, *The Reconfigurable Modular Manipulator System* (RMMS), that addresses the above mentioned shortcomings. It provides a viable alternative to using fixed configuration manipulators by extending the existing concept of modular manipulator design. The term *modular manipulator* generally refers to a robotic manipulator assembled from discrete mechanical joints and links into one of many possible manipulator configurations [17]. Such a manipulator has several advantages over conventional designs, most notably economy of manufacture, ease of modification and ease of repair. At least one such modular manipulator is now commercially available [13].

The Reconfigurable Modular Manipulator System extends the concept of modularity throughout the entire manipulator system to include not only the mechanical hardware, but also the electrical hardware, control algorithms, and software as well. The RMMS (Reconfigurable Modular Manipulator System) utilizes a stock of interchangeable link modules of various lengths, and joint modules of various sizes and performance specifications. This modularity allows a wide range of manipulator architectures to be assembled from a small set of general purpose hardware and software components.

The concept of an RMMS poses challenging technological and theoretical research issues that must be addressed before such a system can be used effectively. In this paper we discuss both theoretical and technological issues and describe our progress in this area. In order to demonstrate our ideas we have built a prototype RMMS in our laboratory. We describe the design and operation of this prototype RMMS. The prototype includes 6 joint and 6 link modules, and a controller consisting of a Motorola 68020 based computer with real-time capabilities. We have also implemented an algorithm that automatically generates forward and reverse manipulator kinematics. The RMMS is presently controlled by independent joint control algorithms. We are now addressing issues such as mapping task specifications to manipulator configurations, automated generation of the manipulator dynamics equations, and reconfigurable model-based control algorithms. Interestingly, a

recent survey indicates a need for manipulators with both reconfigurability and extensibility for research in all areas of robotics [15]. Our RMMS design provides practically all of the features discussed in this survey.

2. Design Philosophy and Implementation

An RMMS consists of similar subsystems as those found in conventional manipulators:

- A physical structure of joints and links.
- Servo systems for each joint, consisting of actuators, transmissions, and sensors.
- A computer controller and programming environment.

The major differences between an RMMS and a conventional manipulator are the standardized component interfaces and configuration independent control algorithms. The interface standardization must include the mechanical mating of manipulator modules, the format of data communication, the communication protocols between hardware and software, and between various levels of software. Although adopting such standards impose some restrictions on the design of the actual components, this disadvantage is offset by the interchangeability of manipulator components and the capability for rapid reconfiguration. In the following subsections, we present the design, and mechanical and electronics interface of each major component in the prototype RMMS system that we have developed in our laboratory.

2.1. Link and Joint Modules

The mechanical modules making up an RMMS are divided into two groups, joints and links. The design of each module is independent of other modules except for the module interfaces which are standardized. One implication of this modular joint design is that the *entire* joint actuator must be packaged *within* the joint module. Each joint module must include a motor (or some type of actuator), a transmission mechanism, a position sensor, and the necessary power electronics to control the motor. Electrical power is distributed and communication is multiplexed over a small number of conductors permanently installed in each module. This allows for simple assembly without custom cabling. Although these design constraints limit the power which can be generated by the joint due to the limited size of the motor, transmission, and power amplifier, this is not viewed as a major short coming of the design. By properly selecting the transmission reduction ratio, high torques at low speeds can be obtained, appropriate for most tasks as long as speed of operation is not critical.

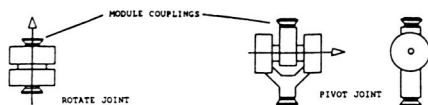


Figure 2-1: Modular Joint Assemblies

For simplicity and convenience, we have considered and built only the two common types of revolute joint in our RMMS. These two types are rotate, and pivot, and are distinguished by the orientation of the joints link axes with the joint axis. Both types of joint are shown schematically in Figure 2-1. A rotate type joint has link axes which are co-linear with each other and with the joint axis. A pivot has link axes which are both perpendicular to the joint axis.

Our current designs for pivot and rotate joints are shown in the photographs in Figures 2-2 and 2-3. The actuator in each joint consists of a conventional servo motor and linear amplifier driving a harmonic drive with 200:1 reduction ratio. This design yields a maximum output torque of 200 ft-lbf, and maximum axis speed of 0.7 radian/second. Also integral with the joint assembly is a brushless resolver mounted coaxially with the output shaft, providing position feedback with a resolution of 0.0001 radians. A wire windup allows the resolver (and output shaft) to turn up to 480° before damaging the resolver electrical connections. In our design we have also allowed for incorporating a tachometer that is directly coupled to the motor shaft. The tachometer will provide output shaft velocity measurements with a resolution of 0.001 radians/second. All of the actuator components are packaged in a sub-assembly of the joint module, allowing a number of kinematically different types of module to be manufactured from this common assembly. The total weight of both types of joint is 17 lbs.

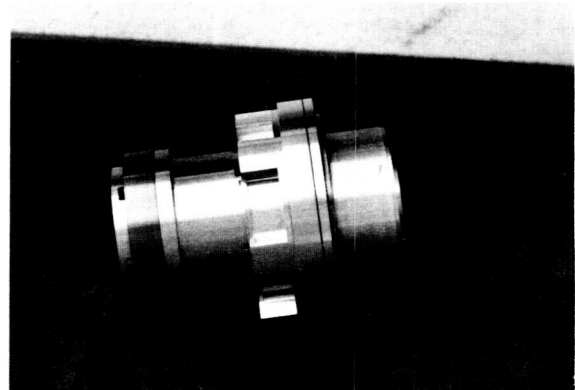


Figure 2-3: Photo of CMU RMMS Prototype Rotate Joint



Figure 2-2: Photo of CMU RMMS Prototype Pivot Joint

We tested the joint modules using a fixed gain, PSD feedback control algorithm. The control loop gains and sampling rate were determined by an experimental procedure [6]. In our experiments, we obtained static positioning accuracies of ± 0.001 radians, and closed-loop stability of the system was demonstrated at sampling rates as low as 100 Hz. We are currently developing techniques for dynamics identification to evaluate the use of model-based reconfigurable controllers for the RMMS.

2.2. Joint - Link Interface

In order to assemble the joint and link modules into a manipulator, a method of mechanically coupling the modules is required. This coupling must both align the modules, and lock them together with sufficient strength to transmit the internal forces generated by the movement of the manipulator. In addition to structurally coupling the modules together, this interface must also electrically couple the modules, and be able to sense the coupling orientation of successive modules.

The current interface design is shown in the photograph in Figure 2-4. The mechanical coupling is accomplished using commercial V-band clamps. V-band flanges are an integral part of the link and joint modules, as shown in Figure 2-2 and 2-4. An arrangement of pins and holes in each flange limits the coupling orientation to four, equally spaced positions that are 90 degrees apart. An LED in one flange and four phototransistors in the other allow the controller to sense which of the four possible orientations is in use. Although rudimentary, this design provides the necessary functionality for the module interface. We are currently investigating the use of quick release V-band clamps and more sophisticated designs with locking mechanisms that allow automatic "peg-in-hole" type coupling.

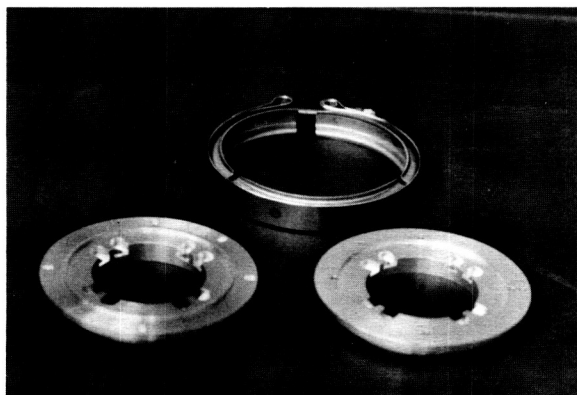


Figure 2-4: Photo of Prototype Module Interface

2.3. Communication Interface

Each joint houses the power and sensor electronics for the actuator. To control the joint actuators and obtain sensor feedback, a communication link between the joint modules and a computer controller is required. To allow standard connections between joint modules, this communication link must be implemented using a fixed number of conductors while being capable of supporting an arbitrary number of modules. This implies a multiplexed communication link, similar to a computer bus or Local Area Network (LAN).

Due to the data transmission overhead associated with existing LANs, our prototype utilizes a bus type implementation, referred to as the *armbus*. The *armbus* design is shown schematically in Figure 2-5. This design is based on a conventional 8-bit bi-directional data/address bus, an additional 5 control lines, and a rather unconventional 4 bit daisy chained node address bus. The daisy chained address bus provides automatic node address configuration; the first module in the manipulator is node address 1, the second module is node address 2, and so on. This is accomplished by including a "subtract one" circuit in each module which is in the path of the node address lines. Each joint can thus detect "address equals zero" as the node address. Due to the low data rate of the bus (current bus clock is 500 KHz), the propagation delay added by the subtract circuit is negligible.

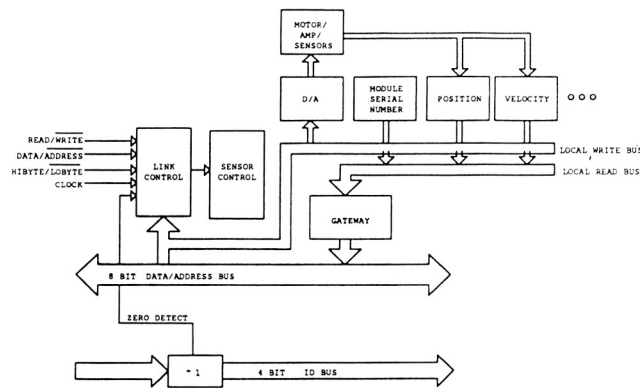


Figure 2-5: Manipulator Communication Bus Logic

2.4. RMMS Computing Environment

RMMS software is easily divided into two functional classes: real-time critical control programs and event-driven application programs. Real-time programs are those which must be executed at a predetermined sampling rate, such as control law calculation. In contrast, event-driven programs rely on detecting conditions, such as the manipulator reaching a certain position, to schedule future manipulator actions. In our implementation, we have chosen this distinction (between real-time and event driven programs) as a natural module boundary for organizing the manipulator control software.

In the RMMS environment, a CPU is dedicated to each class of software. Real-time control programs execute on a dedicated *controller CPU*, with a hardware interface to the inter-module communication network. This controller CPU performs the necessary realtime control of the manipulator, and receive commands from a second, *master CPU*. This master CPU executes the event-driven application program. In this architecture the manipulator controller appears as a peripheral device. An interrupt driven communication channel between the two processors provides a well defined interface between the two software/computing modules.

We have implemented this architecture, depicted in Figure 2-6, for controlling the RMMS. The controller CPU is an Ironics single-board computer, based on a Motorola 68020 processor and VME bus, with 1 MByte of dual ported RAM. The master CPU is a SUN-3 workstation, also based on the Motorola 68020 and VME bus. This basic architecture (and the support software) can be expanded to include additional Ironics CPUs for greater computational power. The similarity between the Ironics and SUN's CPU allows us to use the same editor and compiler for both processors thus simplifying software development and inter-processor communication. Real-time control programs, at all levels, are written entirely in C programming language. The interface to the manipulator communication network is via the VMX bus interface included on the Ironics. The VMX bus is a recognized extension to the VME bus and is intended to be a local IO bus in multiprocessor systems such as the one we have built for controlling the RMMS.

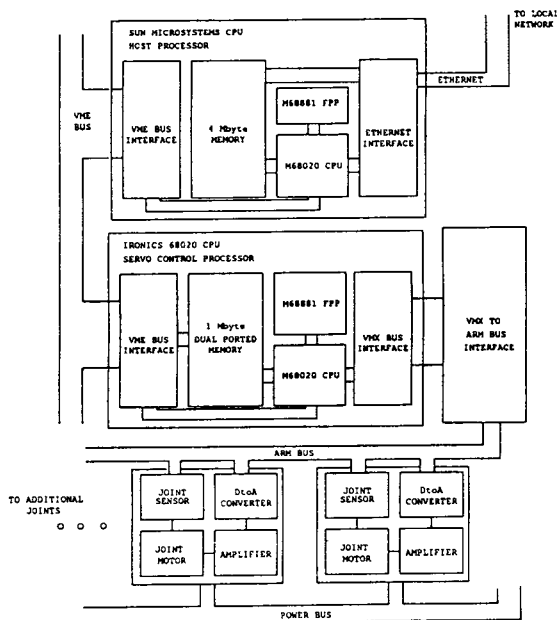


Figure 2-6: Schematic of RMMS Computing Architecture

2.4.1. Real-Time Operating System

Manipulator control programs executing on the Ironics real-time CPU are linked with a locally developed real-time operating system or kernel. This kernel provides a number of concurrency and scheduling primitives, allowing users to write control programs as a series of concurrent processes. It also supports many Unix-like utilities, particularly memory allocation and access to the SUN file system. These features have two important implications to the development of manipulator control code:

- Control algorithms are written without regard to the specific hardware and low level software implementation of the system. At the same time, the programmer is forced to more fully understand the data flow and timing relationships of the algorithm being coded, to specify those relationships via the concurrency primitives.
- By providing real-time programming utilities that mimic their Unix counterparts, a large base of existing Unix/C code is easily ported to real-time applications. Similarly, a large base of existing Unix/C programming expertise is also readily available.

2.4.2. Real-Time Software Architecture

The current software control architecture is shown in Figure 2-7. In the current design there are four principal processes executing concurrently:

- The feedback control law which is implemented for each manipulator axis can be executed at sampling rates of 50-500 Hz. Our current implementation employs a sampling rate of 200 Hz.
- The path planning algorithm updates the control loop inputs to drive the manipulator to a desired position in a specified manner (eg. straight line, minimum time, etc). This can operate at sampling rates of 5-30 Hz. We are presently using a sampling rate of 20 Hz.
- A data logging process that records specified values of the manipulator state. This information is required for off-line analysis and for monitoring manipulator control experiments.
- An interactive command interpreter that implements a low level manipulator control language. This allows a user or an application program on the SUN-3 to issue commands, to the control package, for displaying data about the manipulator state.

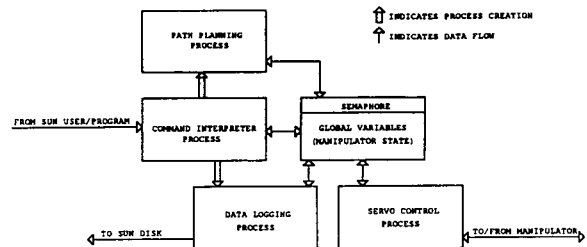


Figure 2-7: Control Software Organization

2.4.3. Real-Time Computing Performance

The Motorola 68020/68881 CPU has been extensively benchmarked for many applications, with typically reported performances of 2 MIPS and 0.25 MFLOPS [14, 8]. In order to determine the performance of the actual system executing a typical

manipulator control program, the RMMS realtime CPU was benchmarked performing a single iteration of a PSD position control loop. The control law calculation is given by the following pseudo-C code. All variables are double precision floating point variables, referenced indirectly by an offset from an address register (the benchmark thus includes a typical level of addressing overhead). The actual code was written with no attempt at optimization other than that performed by the compiler.

```
pos_error = reference_position - position;
vel_error = reference_velocity - velocity;
integral = (integral * alpha) + pos_error;
torque_command = (pos_error * Kp) +
    (vel_error * Kv) + (integral * Ki);
if (torque_command > Tlim)
    torque_command = Tlim;
else if (torque_command < -Tlim)
    torque_command = -Tlim;
```

This computation requires 11 floating point operations (4 multiplies, 5 additions/subtractions, and 2 comparisons). The actual code is fairly typical of first pass code written by an average C programmer. This segment executes in 0.12 milliseconds, indicating floating point performance of approximately 0.1 MFLOPS. Obviously this is a rough measurement of system performance, however this is quite good considering the unoptimized nature of the code. With simple code optimization, it is quite possible that compiled C code could approach the 0.25 MFLOP performance claim.

2.4.4. Application Control Software

Within the RMMS computing environment, application programs are SUN-3 programs, written in a SUN supported language. Currently, we are using the C programming language for developing application programs. Access to the manipulator controller is via special *Unix devices* which implement *pipe* like communication channels to the real-time program. This mechanism has been used to build a message passing protocol between the two processors. This has been done for the existing manipulator control package, allowing a SUN program to call an appropriate library routine which signals the manipulator control program to execute the desired command.

3. Automatic Kinematics Generation

Specifying a manipulator task typically requires specifying the end effector position (with reference to the manipulator base) as a function of time and system conditions. This method of task specification is well suited to an RMMS, as it is completely independent of the manipulator configuration; the manipulator is simply considered a motion transducer. Since the end effector position is controlled indirectly by controlling each joint's axis position, the relationship between these two quantities, known as the manipulator forward and reverse kinematics, is required. Deriving a set of Denavit-Hartenberg parameters (for the forward kinematics) and a closed-form reverse kinematics solution requires both mathematical manipulation and geometric intuition [11]. Further, since an arbitrary manipulator may be created from the RMMS, the forward and reverse kinematics solutions have to be derived for each configuration of the manipulator.

To alleviate the above difficulty we have proposed algorithms that create the forward and reverse kinematics solutions automatically from a description of the joint and link modules and the sequence in which they have been connected. For the reverse kinematics we have adopted a numerical approach that allows for complete generality and can also accommodate redundant manipulators. A general numerical solution to the reverse kinematics is often computationally inefficient and mathematically poorly behaved especially close to singularities. To address this issue, we have developed a robust reverse kinematics solution that is well behaved close to a singularity and can be computed at real-time rates. In the ensuing paragraphs we present our approach to generating the kinematics of a RMMS automatically.

3.1. Generating the Forward Kinematics

The forward kinematic equations of a manipulator describe the position and orientation of the end-effector as a function of the joint variables. The forward kinematic transformation is typically obtained from a set of parameters known as the Denavit-Hartenberg (D-H) parameters of the manipulator. These parameters are obtained through a predefined sequence of transformations and are a function of the geometry of the manipulator. The input to our forward kinematics algorithm is the geometry of each module, the type of each module, and the sequence of connection of the modules that comprise the manipulator. The output of our forward kinematics algorithm is the set of D-H parameters of the manipulator.

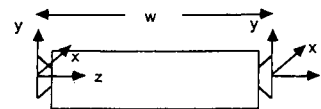


Figure 3-1: Link Module Coordinate Assignment

We use homogeneous transformation matrices to specify the geometry of modules. For a link module we use one homogeneous transformation that relates one end of the link to the other as depicted in Figure 3-1. In order to incorporate both the degree-of-freedom of a joint and its shape we use two homogeneous transformations: one from the lower left connector to the origin of the joint (LJ_o) and another from the origin to the upper right connector (OJ_u). A typical joint module and its database description is shown in Figure 3-2. The definition of the origin of the joint module is arbitrary as long as it is chosen to be a point lying along the axis of rotation. Based on the above systematic description, we have implemented an algorithm that automatically creates the forward kinematics of an RMMS. For the sake of brevity we have excluded the details of the algorithm in this paper, they are presented in [4].

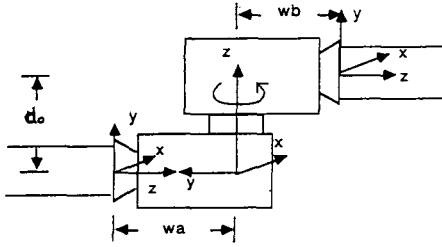


Figure 3-2: Joint Module Coordinate Assignments

A simpler method of generating the forward kinematics of an RMMS would be to sequentially multiply all the module transformations. However, it is desirable (particularly when the manipulator Jacobian is also required) to represent the forward kinematics in terms of the Denavit-Hartenberg parameters. In the present implementation, the control computer reads the description of the joint and link module descriptions through a database file. However, in the future each joint and link module will have a ROM which will include the kinematic information pertaining to that module.

3.2. Reverse Kinematics of RMMS

In order to do any controlled movement it is necessary to have an inverse kinematic model to determine the joint angles required to achieve a desired position and orientation of the end-effector. Ideally, one derives closed form equations for the inverse kinematics where each joint variable is expressed in terms of other known quantities. However, existence of a closed form inverse kinematics solution depends on the kinematic structure of the manipulator [12, 16]. For example, it is known that a closed form solution exists for a manipulator which has three consecutive axes that intersect, such as in a spherical wrist [12]. This solvability condition is not necessary, but only sufficient. Because an RMMS manipulator can assume any configuration, including one that is redundant, it may not be possible to find a closed form solution. In order to provide for generality we have adopted a numerical approach for solving the inverse kinematics of an RMMS. In the ensuing paragraphs we describe a numerical method to compute the inverse kinematics of non-redundant manipulators [5]. We also describe an extension of this method that is applicable for redundant manipulators.

3.2.1. Inverse Kinematics of Non-Redundant Manipulators

A closed-loop method for solving the inverse kinematics equations using the Newton Raphson method is proposed in [5] and is depicted in Block diagram form in Figure 3-3. The iterative method determines the necessary changes in the joint angles to achieve a differential change in the position and orientation of the end-effector. The forward kinematics are described in functional form as:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}). \quad (1)$$

where \mathbf{x} is the vector of Cartesian position and orientation and \mathbf{q} is a vector of joint displacements. The corresponding differential changes $d\mathbf{x}$ and $d\mathbf{q}$, in the Cartesian and joint space, respectively, are

related through the manipulator Jacobian as:

$$d\mathbf{x} = \mathbf{J}(\mathbf{q})d\mathbf{q}. \quad (2)$$

Inverting Equation (2) to obtain an expression for the differential inverse kinematics we obtain:

$$d\mathbf{q} = \mathbf{J}^{-1}(\mathbf{q})d\mathbf{x} \quad (3)$$

where \mathbf{J}^{-1} is the inverse Jacobian. The above equation may be written, in an iterative form, as:

$$d\mathbf{q}_{k+1} = \mathbf{J}^{-1}(\mathbf{q}_k)d\mathbf{x}_k \quad (4)$$

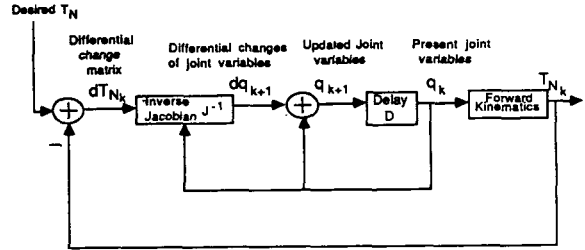


Figure 3-3: Block Diagram of Inverse Kinematics Algorithm

where the differential change in position and orientation at the k -th iteration is computed from the differential homogeneous transformation matrix $d\mathbf{T}_{N_k}$ [11]. The joint displacements are computed as:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + d\mathbf{q}_{k+1}$$

Equation (4) is solved iteratively, until each term in \mathbf{T}_{N_k} (or correspondingly in $d\mathbf{x}_k$) is within a prespecified error tolerance, ϵ .

We have performed experiments using the above algorithm and have shown it to work well for non-redundant systems. Including redundancy introduces complications in the computation of the inverse kinematics solution. The Jacobian, which relates differential changes in the joint variables to differential changes in the Cartesian variables is of dimension $M \times N$, where M is the number of degrees of freedom of the workspace and N is the number of degrees of freedom in the manipulator. When M and N are not equal (which is the case for redundant manipulators), the Jacobian is no longer invertible and we must substitute a generalized inverse to provide an inverse equivalent.

Much of the previous research on inverse kinematics for redundant manipulators has focused on the pseudoinverse [1, 3, 7]. The pseudoinverse is a generalized inverse which provides the minimum norm solution [10]. Because standard pseudoinverse control has proved to be inadequate in the neighborhood of singularities, many methods have been developed which augment the pseudoinverse so as to use the kinematic redundancy to optimize an objective function [1, 2, 7].

While methods cited above are configuration dependent, computationally intensive, or both, the method we propose for RMMS achieves singularity avoidance while requiring negligibly more computations than the standard pseudoinverse. It is called the singularity robust inverse [9]. The pseudoinverse solution is

problematic in the neighborhood of a singularity. In an effort to converge to an exact solution, the pseudoinverse may generate an infeasible solution. That is, it may generate a solution for which one, or more, of the dq values is so large that it cannot be physically realized. The singularity robust inverse method circumvents this problem by providing continuous and feasible solutions even at, or in the neighborhood of, singular points.

The singularity robust inverse is based upon an evaluation index,

$$d\phi = \begin{pmatrix} dx - J \cdot dq \\ dq \end{pmatrix}, \quad (5)$$

which simultaneously considers the exactness of the solution, as measured by the top term, and the feasibility of the solution, as measured by the bottom term. When solving the inverse kinematics problem one must find the minimum weighted Euclidean norm of the evaluation index. The weighting of the terms in the evaluation index manifests itself with the scale factor λ . The singularity robust inverse, J^* becomes:

$$J^* = J^T(J^T J + \lambda I)^{-1}. \quad (6)$$

In the next section we discuss a technique for choosing the parameter λ .

3.3. A Method for Choosing the Scale Factor

In order to employ the singularity robust inverse for RMMS, we must develop a method to automatically generate an appropriate scale factor for any manipulator. The scale factor, λ , must have a large value in the neighborhood of singular points and must be small value, or zero, far from singular points. This is achieved by computing λ as [9]:

$$\lambda = \begin{cases} \lambda_0(1 - \frac{\omega}{\omega_0}) & \text{if } \omega < \omega_0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $\omega = \sqrt{\text{determinant}(J \cdot J^T)}$ is a manipulability measure for the manipulator [18], λ_0 is the magnitude of the scale factor at singular points, and ω_0 is a threshold which represents the neighborhood of singular points. Equation (7) automatically adjusts λ according to the manipulator's distance from a singular point.

To experimentally implement the above method it is necessary to choose values for the parameters λ_0 and ω_0 . Further, the choice of these parameters must be configuration independent and work without a priori knowledge of the location of manipulator's singularities or kinematic parameters. While the value of ω approaches zero as the manipulator approaches a singular point, its absolute magnitude is dependent on the dimensions and the units of measure of the links and joints of the manipulator. For example, an ω of 10^2 may imply that one manipulator is near a singular point, but another manipulator, which has much smaller dimensions, may be far from one. In order to remove the dependency of ω on the units of measure and the absolute values of the kinematic lengths, we have introduced the idea of a scaling a manipulator. Scaling is accomplished by dividing all the kinematic lengths by the largest length of a manipulator. This forces all the kinematic lengths to lie

between zero and one thus diminishing the disparity in the magnitudes of ω between different manipulators. However, different scaled manipulators may still generate vastly different ω values.

The singularity robust inverse chooses an absolute threshold value to specify ω_0 . As mentioned before this choice is manipulator dependent. In order to alleviate this difficulty, we propose checking for a sudden drop in the value of ω between iterations. This is motivated by the observation that as a manipulator approaches singular configuration the value of ω decreases dramatically. We detect the neighborhood of a singularity when the ratio $\frac{\omega_{k+1}}{\omega_k}$ falls below a threshold μ . That is, we examine the ratio of ω between the k^{th} and the $k+1^{th}$ iterations of the Newton-Raphson algorithm.

Based upon the above discussion, the equation for computing the scale factor λ (for a scaled manipulator) is:

$$\lambda = \begin{cases} \lambda_0(1 - \frac{\omega_{k+1}}{\omega_k}) & \text{if } \frac{\omega_{k+1}}{\omega_k} < \mu \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Our experiments with the above technique suggest $\mu \approx 0.1$ to be reasonable value.

We choose λ_0 based on the tradeoff that is the premise for the singularity robust inverse method. Namely, by adding a larger scale factor we make the solution less exact, but more feasible or robust. In order to generate a less exact solution we must increase ϵ . (Recall ϵ is the convergence error tolerance for the Newton-Raphson algorithm.) While increased error tolerance is acceptable for many applications, we cannot assume so for the general case. Alternatively, we maintain the error tolerance and increase the number of iterations of the Newton-Raphson algorithm until the error is less than ϵ .

Before choosing a value for λ_0 we must determine how large λ can be before the system fails to converge. In order for the Newton-Raphson iteration to converge, the residual error must be less than the error tolerance ϵ . Therefore, λ must be also be less than ϵ . Rather than defining an absolute value for λ_0 , we propose setting λ_0 equal to one order of magnitude smaller than ϵ ($\lambda_0 = 0.1\epsilon$). This choice is based upon our experimental results with $\mu = 0.1$.

4. Summary

In this paper we have describe the design of an RMMS. The feasibility of such a system has been demonstrated through the construction of a prototype RMMS built using readily available commercial components. A powerful computer control system with both real-time scheduling and Unix compatibility has also been built, and used to control the current RMMS manipulator.

As part of the effort to develop reconfigurable control programs, an algorithm for automatic forward and reverse kinematics generation has been implemented and tested. The algorithm is implemented as a computer program, which can find the Denavit-Hartenberg parameters for an arbitrary configuration manipulator,

and then perform an iterative inverse kinematics solution. The inverse kinematics algorithm has been extended to work for redundant manipulators. The extended algorithm generates manipulator solutions which avoid singular positions. Both algorithms have been optimized for computational efficiency and robustness, and have been implemented on an Motorola 68020/68881 based single board computer, at rates on the order of 20 Hz.

References

- [1] J. Baillieul, J. Hollerbach, R. Brockett.
Programming and Control of Kinematically Redundant Manipulators.
Proc. 23rd Conference on Decision and Control :768-774, December, 1984.
- [2] J. Baillieul.
Kinematic Programming Alternatives for Redundant Manipulators.
IEEE International Conference on Robotics and Automation 1:722-728, March, 1985.
- [3] P.H. Chang.
A Closed-form Solution for the Control of Manipulators with Kinematic Redundancy.
IEEE International Conference on Robotics and Automation 1:9-14, April, 1986.
- [4] L. Kelmar and P. Khosla.
Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System.
In IEEE Conference on Robotics and Automation. IEEE, April, 1988.
- [5] P.K. Khosla, C.P. Neuman, and F.B. Prinz.
An Algorithm for Seam Tracking Applications.
The International Journal of Robotics Research 4(1):27-41, Spring, 1985.
- [6] Khosla, P. K.
Real-Time Control and Identification of Direct-Drive Manipulators.
PhD thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University, August, 1986.
- [7] C. A. Klein and C-H Huang.
Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators.
IEEE Trans. on Systems, Man, and Cybernetics SMC-13(3):245-250, March/April, 1983.
- [8] Beims, Bob.
The Floating-Point Performance Standard Gets Even Faster!
In (editor), *WESCON 1986 Professional Program Papers*, Session 35/1. Electronic Conventions, 1986.
- [9] Y. Nakamura and H. Hanafusa.
Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control.
Journal of Dynamic Systems, Measurement, and Control 108:163-171, September, 1986.
- [10] B. Noble and J.W. Daniel.
Applied Linear Algebra.
Prentice Hall, N.J., 1977.
Second Edition.
- [11] Paul, R. P.
Robot Manipulators : Mathematics, Programming and Control.
MIT Press, Cambridge, MA, 1981.
- [12] Pieper, D. L.
The Kinematics of Manipulators under Computer Control.
PhD thesis, Department of Computer Science, Stanford University, 1968.
- [13] Anon.
Technical Brochure on Modular Arms
Robotics Research Inc., Ohio, 1987.
- [14] Anon.
The SUN-3 Family: An Overview
SUN Microsystems Inc., California, 1986.
- [15] Walker, M.
A Survey of Research Robots.
Technical Report, University of Michigan, Ann Arbor, 1987.
- [16] W.A. Wolovich.
Robotics: Basic Analysis and Design.
Holt, Rinehart and Winston, New York, 1987.
- [17] Wurst, K. H.
The Conception and Construction of a Modular Robot System.
In Proceedings of the 16-th International Symposium on Industrial Robotics, pages 37-44. ISIR, 1986.
- [18] Yoshikawa, T.
Manipulability of Robotic Mechanisms.
In Proceedings of the Second International Symposium on Robotics Research. MIT, Kyoto, Japan, August 20-23, 1985.

A ROBUST CONTROL SCHEME FOR FLEXIBLE ARMS WITH FRICTION IN THE JOINTS

Kuldip S. Rattan¹, Vicente Feliu², and H. Benjamin Brown Jr.

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

A general control scheme to control flexible arms with friction in the joints is proposed in this paper. This scheme presents the advantage of being robust in the sense that it minimizes the effects of the Coulomb friction existing in the motor and the effects of changes in the dynamic friction coefficient. A justification of the robustness properties of the scheme is given in terms of the sensitivity analysis.

1 INTRODUCTION

In the last few years, considerable research effort has been devoted in controlling flexible structures and, in particular, flexible arms [1-6]. Very little effort has been made in controlling flexible arms when Coulomb and dynamic friction are present in the joints, in spite of this being common in practice. In fact we have not found any paper that specifically deals with and tries to solve this problem.

This paper is devoted to solving the above problem. In order to do this, a new general control scheme is proposed. Existing methods to control flexible arms are based on the explicit control of the tip position, where the controller generates the current for the D.C. motor of the joint as a control signal. The proposed method is based on the simultaneous control of the joint motor position and tip position, and the implementation of two nested closed loops: the inner loop that controls the motor position and the outer loop that controls the tip position.

The general control scheme is presented in Section 2. Section 3 compares the sensitivity of our general control scheme with the sensitivity of other existing methods. This shows our scheme to be more robust to changes in friction than the others. Finally, some conclusions are drawn in Section 4.

2 GENERAL CONTROL SCHEME

As was mentioned in the introduction, there are many applications in which friction must be taken into account when controlling a flexible arm. Only in the case when the coupling torque between the beam and the motor is many times larger than the friction torque it can be neglected. This may be true for very large structures, but it is not true in many industrial applications.

In order to reduce the effects of friction, the control scheme of figure 1 is proposed. In this scheme two variables are controlled: motor and tip position (θ_m and θ_t respectively). These two variables are controlled by two nested closed loops and two different controllers ($R_1(s)$ and $R_2(s)$) are used. These controllers are each designed separately and according to different criteria. In figure 1, the open-loop transfer function of the motor $G_M(s)$ has all its poles and zeroes in the left half-plane. The open-loop transfer function of the flexible beam $G_B(s)$ has its poles in the left half-plane but may have (for arms with more than one vibrational mode) some zeroes in the right half-plane (non-minimum phase system). $F(s)$ is a filter for the reference and is normally designed

¹Visiting Professor, Department of Electrical System Engineering, Wright State University, Dayton, OH. 45435.

²Visiting Professor, Dpto Ingenieria Electrica y Control, UNED, Ciudad Universitaria, Madrid-28040, Spain.

in conjunction with $R_1(s)$.

The use of this control scheme has been motivated by the well known property that the sensitivity of a closed-loop system to perturbations can be made arbitrarily small by increasing the gain of the open-loop, provided that the system remains stable (Kuo [8]). Therefore considering the friction (in general terms) as a perturbation, we can reduce its effects by increasing the gains of the controller.

If we try to do this using the existing control schemes (such as shown in figure 2) for flexible arms, the gains cannot be arbitrarily increased because of the right half-plane zeroes. In the proposed scheme, because $G_M(s)$ is minimum phase, the gains of the inner loop can be arbitrarily increased (using an appropriate controller $R_2(s)$) without making the system unstable. Intuitively, the high gain inner loop to control the motor position makes the system insensitive to friction and, a second outer loop may then be designed to control the tip position. This second loop cannot have a high gain because $G_B(s)$ is non-minimum phase. However, the friction effects have been largely removed by the high-gain inner loop controlling the motor position.

3 SENSITIVITY ANALYSIS

The previous ideas intuitively justify the reason for using our control scheme. This section is devoted to giving an analytical proof. The analysis carried out here is quite straightforward and will give a quantitative idea of how much the robustness is increased using the nested loop scheme. In order to do this comparison, a typical control scheme like the one shown in figure 2 will be used (Cannon [1]). The sensitivity characteristics of this system will be taken as representative of the existing methods because they are based on controlling the tip position using a controller that generates a command for the current of the D.C. motor. The sensitivities of these methods are thus of the same order of magnitude. Two comparative analyses will be done : one checking the signal-to-noise ratio (considering the Coulomb friction as the noise), and the other checking the sensitivity to variations in the dynamic friction coefficient.

In order to do this comparative analysis, the state-space control scheme of figure 2 is first expressed in terms of its equivalent transfer functions.

Assume that the plant is represented by the state space-equations

$$\begin{aligned}\dot{\underline{x}} &= A\underline{x} + Bi \\ \theta_t &= C\underline{x}\end{aligned}\quad (1)$$

where $C = [1 \ 0 \ 0 \ 0 \ \dots \ 0]$ is of dimension n . The controller is a row vector of dimension n of the form

$$\underline{K}^T = [k_1 \mid \hat{\underline{K}}^T] \quad (2)$$

where $\hat{\underline{K}}$ is a vector of dimension $n-1$, and k_1 is the coefficient corresponding to the tip position error (error in the first state). The states \underline{x} of the system can be reconstructed from the state-space equation (1) as follows

$$\begin{pmatrix} \theta_t \\ \dot{\theta}_t \\ \ddot{\theta}_t \\ \vdots \\ \theta_t^{(n-1)} \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ CB & 0 & 0 & \dots & 0 & 0 \\ CAB & CB & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ CA^{n-2}B & CA^{n-3}B & CA^{n-4}B & \dots & CB & 0 \end{pmatrix} \begin{pmatrix} i \\ \dot{i} \\ \ddot{i} \\ \vdots \\ i^{(n-1)} \end{pmatrix} \quad (3)$$

where $[C \ CA \ CA^2 \ \dots \ CA^{n-1}]$ is the observability matrix. If the system is observable (which is true in all the models of flexible arms), then the states \underline{x} of the system may be reconstructed from the measurements of the input i and the output θ_t using a linear law of the form

$$\underline{x}(s) = P(s)\theta_t(s) + Q(s)i(s) \quad (4)$$

where P and Q are polynomial column vectors in s . Equation (4) can be easily obtained from equation (3) by inverting the observability matrix. Adding equation (4) in the scheme of figure 2 and substituting the state-space equation (1) of the plant by $G_M(s)G_B(s)$, we obtain the system shown in figure 3. Figure 3 can be simplified and an equivalent transfer function scheme can be obtained, as shown in figure 4, where

$$F(s) = \frac{k_1 G_M(s)G_B(s)}{KP(s)G_M(s)G_B(s) + KQ(s)} \quad (5)$$

and

$$R(s) = \frac{KP(s)G_M(s)G_B(s) + KQ(s)}{G_M(s)G_B(s)} \quad (6)$$

The comparative analysis may now be done between the schemes of figures 1 and 4.

3.1 Signal-to-Noise Ratio Analysis

The signal-to-noise ratio of a system is defined (Kuo [8]) as

$$R = \frac{\text{Output due to signal}}{\text{Output due to noise}} \quad (7)$$

and is a measure of the sensitivity of the system to perturbation signals (in this case the Coulomb friction). The comparison of the ratios of the schemes of figures 1 and 4 is done here for the same levels of input θ_t^r and perturbation p (which is added to the current of the motor).

The output of the nested double loop scheme in terms of the reference input θ_t^r and the noise p can be written from figure 1 as

$$\theta_t = \frac{G_M(s)G_B(s)}{1 + G_M(s)R_2(s)(1 + G_B(s)R_1(s))} (R_1(s)R_2(s)F(s)\theta_t^r + p) \quad (8)$$

The signal-to-noise ratio for this scheme is given by

$$R_{\theta_t^r, p} = R_1(s)R_2(s)F(s) \quad (9)$$

Similarly the output and the signal-to-noise ratio for the system shown in figure 4 can be written as

$$\theta_t = \frac{G_M(s)G_B(s)}{1 + G_B(s)G_M(s)R(s)} (R(s)F(s)\theta_t^r + p) \quad (10)$$

$$R_{\theta_t^r, p} = R(s)F(s) = k_1 \quad (11)$$

Comparing both results, expression (9) can be made large by designing the controller $R_2(s)$ with arbitrarily high gain because the inner loop is minimum phase. However, k_1 in expression (11), and all the parameters of $R(s)$ in general, are limited by the stability margin since the system is non-minimum phase. The gains of $R_1(s)$ in equation (9) are also limited for the same reason.

From all this it follows that, in general, (9) may be made larger than (11) by properly choosing the gains of the controller of the inner loop. It must be mentioned here that this is a theoretical analysis. In practice, the gains of the inner loop will be limited by the saturation of the amplifier, instability because of unmodelled high frequency dynamics, or even instability because of the discretization of the signals when using digital controllers. But in any case these limits are much larger than the ones imposed by the non-minimum phase characteristic.

3.2 Sensitivity to the Dynamic Friction Coefficient

Dynamic friction is the second manifestation of friction. This is normally assumed to be linear and is included in the model of the plant. In many cases, however, this assumption of linearity is not correct. Often, the dynamic friction coefficient changes noticeably depending on the sense of rotation of the motor [9] or on the position of the rotor of the motor relative to the stator (con-

fronted poles), etc. It will be shown here (by performing the sensitivity analysis of both systems to changes in the dynamic friction coefficient) that the robustness of the system due to changes in the dynamic friction coefficient may be significantly improved by using the nested double loop scheme.

The sensitivity to changes in the parameter v of a system whose closed-loop transfer function is $M(s)$, is defined (Kuo [8]) by

$$S_{M,v} = \frac{dM(s)/M(s)}{dv/v} \quad (12)$$

In order to do this analysis, let us express $G_M(s)$ in the form

$$G_M(s) = \frac{K_m}{Js^2 + vs + T(s)} \quad (13)$$

which is the typical transfer function of a D.C. motor, except the term $T(s)$ which represents the coupling torque between the beam and the motor. This allows us to characterize the influence of the dynamic friction coefficient, v , in the general transfer function.

Performing some calculations, we find the sensitivity to v for the closed-loop system of figure 1 is given by

$$S_{M,v1} = \frac{-sv}{[1 + R_2(s)G_M(s)\{1 + R_1(s)G_B(s)\}](Js^2 + vs + T(s))} \quad (14)$$

The sensitivity for the system of figure 4 is

$$S_{M,v2} = \frac{-sv}{(1 + R(s)G_M(s)G_B(s))(Js^2 + vs + T(s))} \quad (15)$$

The ratio of the two sensitivities given by equations (14) and (15) can be written as

$$\frac{S_{M,v1}}{S_{M,v2}} = \frac{1 + G_M(s)[R(s)G_B(s)]}{1 + G_M(s)[R_2(s)(1 + R_1(s)G_B(s))]} \quad (16)$$

The ratio given by equation (16) is significantly smaller than 1 since $R_2(s)(1 + R_1(s)G_B(s)) \gg R(s)G_B(s)$. Notice also that the gains of $R(s)$ and $R_1(s)$ are bounded by a stability margin and will thus have the same order of magnitude; but the gain of $R_2(s)$ may be increased arbitrarily. Consequently, scheme of figure 1 is more robust in general to changes in the dynamic friction than the scheme of figure 4, by a factor of approximately $R_2(s)$.

3.3 Comparison of the Characteristic Equations

The previous analysis gives a quantitative justification of how the robustness of the system is increased using the two nested loops scheme. A qualitative understanding of how the nested loops modify the stability of the system allowing higher gains may be obtained by looking at the characteristic equations of the two systems.

In the traditional control scheme, the robustness depends on $R(s)$. The characteristic equation of the system shown in figure 4 is given by

$$1 + R(s)G_M(s)G_B(s) = 0 \quad (17)$$

Substituting equation (6) in (17), the characteristic equation can be expressed in terms of the feedback gains as

$$1 + \frac{KG_M(s)G_B(s)P(s)}{1 + KQ(s)} = 0 \quad (18)$$

Notice that in equation (18) the right half-plane zeroes of $G_B(s)$ limit the gain, K , and consequently the gains of $R(s)$.

In the proposed scheme, we note from equation (9) that the signal-to-noise robustness depends on the product $R_1(s)R_2(s)$, while from equation (14), that the sensitivity depends both on this product and also on the $R_2(s)$ term. The characteristic equation of the system can now be written as

$$1 + R_2(s)G_M(s)(1 + G_B(s)R_1(s)) = 0 \quad (19)$$

If the factor $1 + G_B(s)R_1(s)$ had all its zeroes in the left half-plane, then the gains of $R_2(s)$ could be made arbitrarily large. Because $G_B(s)$ is stable, there always exists an $R_1(s)$ of moderately low gains that makes the above factor have all its zeroes in the left half-plane (this may be justified from the root locus plot of $G_B(s) R_1(s)$) and, consequently, the product $R_1(s)R_2(s)$ may be arbitrarily large, improving the robustness of the system.

4 CONCLUSIONS

In this paper, a new control scheme to control flexible arms with friction in the joints has been presented. The proposed scheme is based on a nested double loop structure. It has been shown to be more robust to changes in the dynamic friction coefficient as well as less sensitive to the presence of Coulomb friction as compared to existing methods.

The non-minimum phase feature that appears in the most of the flexible arms precludes the use of very high gains in the controller, which is the classical way of increasing the robustness. The proposed scheme divides the control problem into two: the first consists of the design of an inner loop controller for a minimum phase system, and the second consists of the design of an outer loop to control the tip position where the non-minimum phase feature must be considered.

Analytical comparative studies of the sensitivity to both kinds of perturbation have shown that the proposed scheme is more robust to these perturbations than traditional schemes by a ratio of roughly the gain of the controller of the inner loop. This gain may be made arbitrarily high (the only limitation will be the saturation of the amplifier of the motor and the presence of high frequency unmodelled modes).

REFERENCES

1. Cannon, Robert H and Schmitz, E., "Precise Control of Flexible Manipulators", Robotics Research, 1985.
2. Harahima, F. and Ueshiba, T., "Adaptive Control of Flexible Arm using the End-Point Position Sensing", Proceedings Japan-USA Symposium of Flexible Automation, Osaka (Japan), July 1986.
3. Siciliano, B., Yuan, B.S. and Book, W.J., "Model Reference Adaptive Control of a One Link Flexible Arm", 25th IEEE Conference on Decision and Control, Athens, December 1986.
4. Ower, J. C. and Van de Vegte, J., "Classical Control Design for a Flexible Manipulator: Modeling and Control System Design", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 5, October 1987.
5. Rovner, D.M. and Cannon, R.H., "Experiments Towards on-line Identification and Control of a Very Flexible One-Link Manipulator", International Journal of Robotics Research, Vol. 6, No. 4, Winter 1987.
6. Matsuno, F., Fukushima, S. and Coworkers, "Feedback Control of a Flexible Manipulator with a Parallel Drive Mechanism", International Journal of Robotics Research, Vol. 6, No. 4, Winter 1987.
7. Craig, J.J., "Introduction to Robotics, Mechanics & Control", Addison-Wesley Publishing Company, 1986.
8. Kuo, B.C., "Automatic Control Systems", Prentice-Hall, 1982.
9. Feliu, V., Rattan, K.S., and Brown, H.B., "Model Identification of a Single-Link Flexible Manipulator in the Presence of Friction", 19 Annual Modeling and Simulation Conference, Pittsburgh, May 1988.

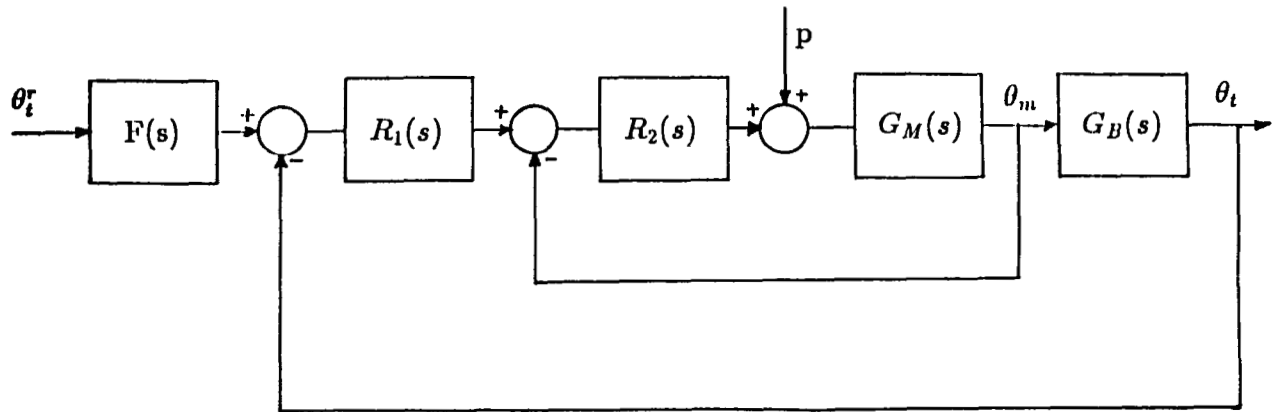


Figure 1. Proposed General Control Scheme.

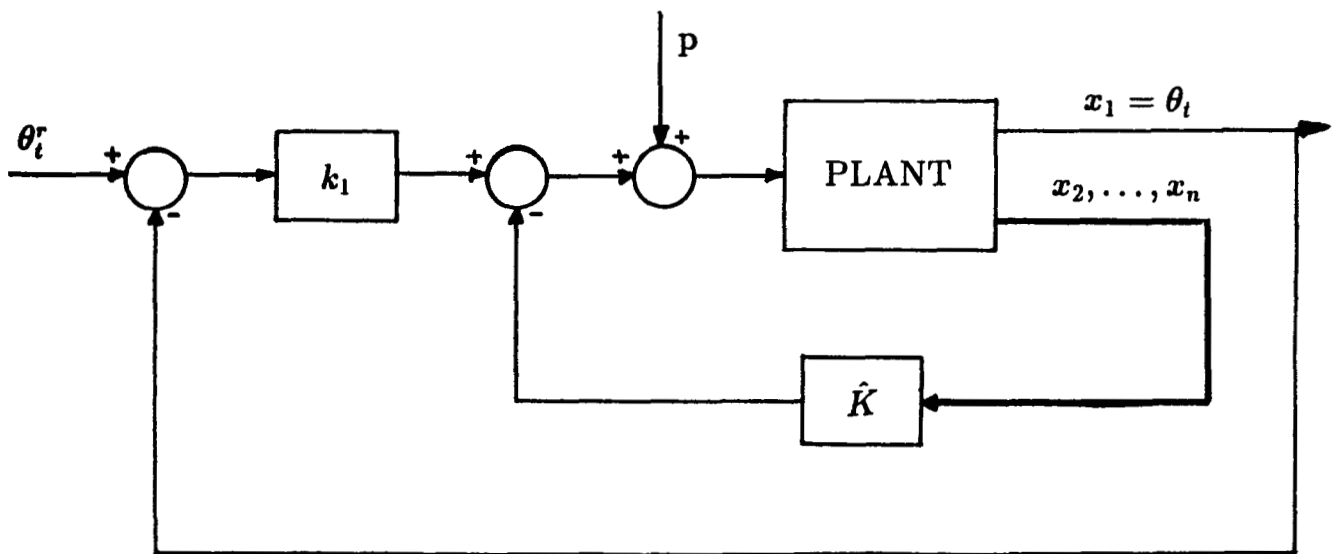


Figure 2. Existing Control Scheme for Flexible Arms.

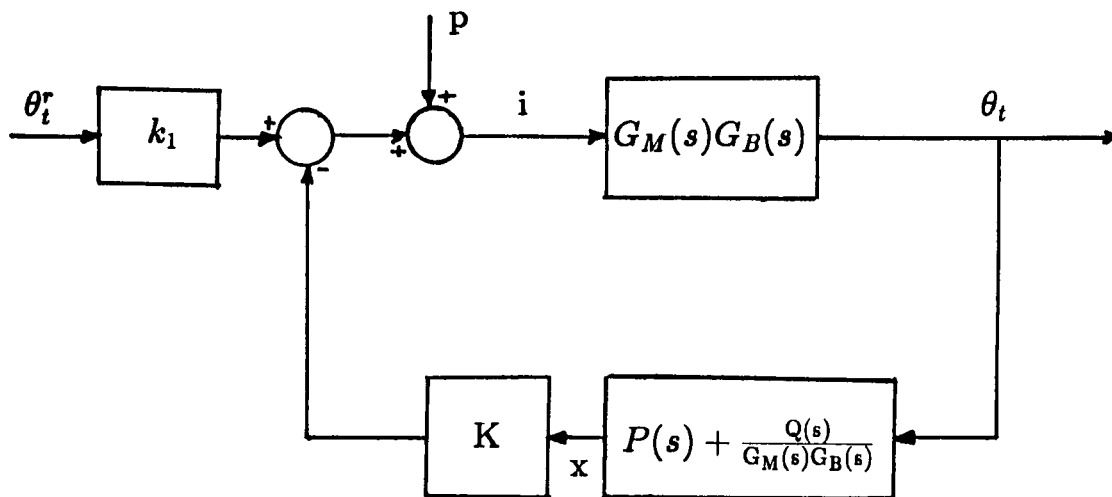


Figure 3. Existing Control Scheme Modified for Output Feedback.

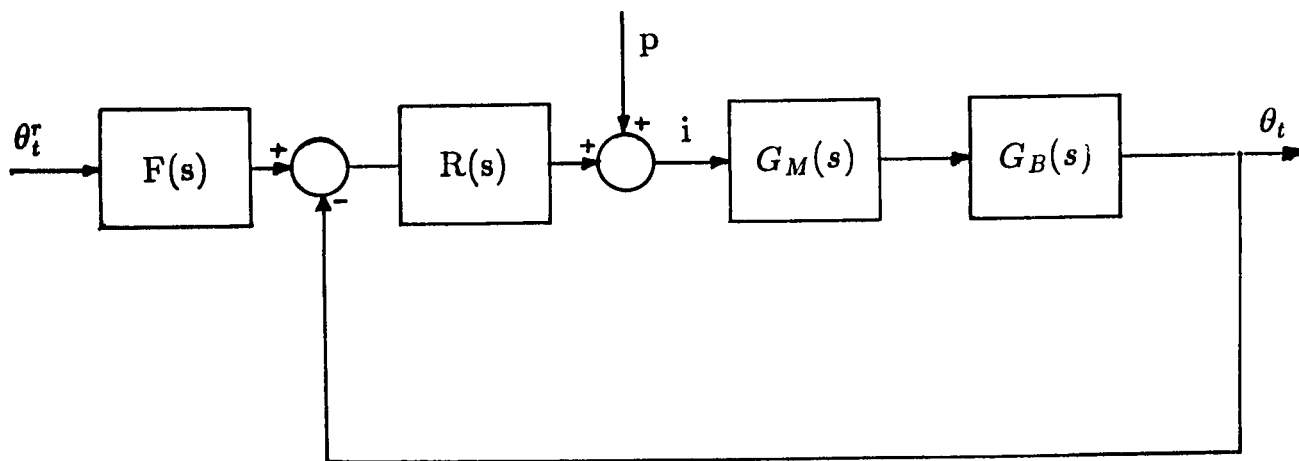


Figure 4. Existing Control Scheme Transformed into Equivalent Transfer Function Form.

TELEROBOT OPERATOR CONTROL STATION REQUIREMENTS

Edwin P. Kan
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, Ca. 91109

ABSTRACT

The Operator Control Station of a Telerobot System has unique functional and human factors requirements. It has to satisfy the needs of a truly interactive and user-friendly complex system; a telerobot system being a hybrid between teleoperated and autonomous system. These functional, hardware and software requirements are discussed in this paper, with explicit reference to the design objectives and constraints of the JPL/NASA Telerobot Demonstrator System.

INTRODUCTION

A telerobot system is a hybrid system between a teleoperated system and an autonomous (robotic) system. It is capable of being teleoperated such as a master-slave manipulator system used widely in the nuclear industry. It is capable of being operated autonomously under preprogrammed control or under real-time intelligent, sensor-based and knowledge-based control, such as in various forms of automated factory systems.

More importantly, a telerobot system is capable of being operated in a continuum mode, where teleoperated control and autonomous control can be traded or shared, as the circumstance requires. That is, when it is more natural and easier to do human teleoperated control, the telerobot system will be configured to do so. When it is more efficient to perform the task autonomously, the telerobot system will be configured to do so. And, when certain degrees of freedom are more readily achieved by autonomous control while other degrees of freedom are naturally achieved by teleoperated control, the overall control is shared between

the two control modes. In all cases, the human operator provides the guidance, direction, high-level decision, supervision and execution of the task.

It is thus apparent that human factors issues are very demanding in the design of the operator control station of this telerobot system. Design requirements are very unique. Based on past experience of operational and research teleoperated and robotic systems, the Jet Propulsion Laboratory (JPL) is now developing a state-of-the-art Operator Control Station (OCS) as part of the NASA (National Aeronautics and Space Administration) Telerobot Demonstrator Project [1]. This paper will present the functional, hardware and software requirements of the OCS of this Telerobot Demonstrator System.

TELEROBOT SYSTEM REQUIREMENTS

The system requirements are driven by the tasks to be performed by the telerobot system; and also driven by the state of the technology. The telerobot system must be able to perform tasks equally as well, and hopefully better, than a straightforward teleoperated system and than a straightforward robotic system. For space applications, the telerobot system must offer advantages over expensive and hazardous EVA (extra-vehicular activities) and IVA (intra-vehicular activities) where astronauts actively participate in the performance of construction, servicing and repair.

In the context of the JPL/NASA Telerobot Demonstrator System, the system is designed as a laboratory testbed for developing, testing and integrating multi-disciplinary technologies. The system is designed within certain overall

constraints of the space environment, while not limited by near-term technology specifics regarding maturity in hardware and software designs.

Overall telerobot system capabilities [2] are targeted as follows:

1. Force reflection in teleoperation.
2. Hybrid position/force control in robotic and advanced teleoperation.
3. Dual arm coordinated control in robotic and teleoperation.
4. Collision prediction, detection and avoidance in robotic operation; use of virtual force field in teleoperation.
5. Supervision by human operator to perform robotic operation and to trade/share between robotic operation and teleoperation.
6. Automatic task planning and command generation: a must for robotic operation, and as a suggestive direction for teleoperation.
7. On-line manipulator path planning: a must for robotic operation, and as an aide for teleoperation.
8. Automatic tracking of labelled and unlabelled objects.
9. Operator initiated object designation and subsequent automatic verification of modeled objects.
10. Simulation of a delay of up to 500 milliseconds (round trip) between the (local) operator's station and the (remote) manipulators.
11. Integrated system operation; ease of operator-machine interaction.

These capabilities are to be demonstrated by performing laboratory tasks simulating those encountered in servicing satellites in orbit. Typically, they include coordinated two-arm manipulation of a large module, an ORU (Orbit Replacement Unit), and include the grappling/halting of a rotating satellite. Dexterous operations in terms of removal of panels, bolts, electrical connectors, tool exchange, object manipulation with precisely defined or loosely defined data bases, are in the list of demonstrations.

OCS AS PART OF THE TELEROBOT SYSTEM

Figure 1 shows the current JPL Telerobot Demonstrator System. It is shown with a generic OCS which is to be replaced by a state-of-the-art OCS in mid-1989.

The Telerobot System has the following subsystems:

1. Operator Control Station (OCS)
2. Artificial Intelligence Planner (AIP)
3. Run-Time Control (RTC)
4. Sensing and Perception (S&P)
5. Manipulators and Control Mechanization (MCM), including Teleop components
6. System Executive (SE)

This above partitioning is driven by the identification and separation of system functions, specific technology, hardware and software. (For more specifics, refer to [2]).

For a user-friendly and machine-interactive operation, the OCS discussed in this paper actually crosses some boundaries in the above system partition. The crossed boundaries include AIP, SE and MCM/Teleoperation components. It will be clear in the following sections that the user interface now placed in AIP, the system management in SE and the hand controller hardware in Teleoperation subsystems are truly part of an overall Operator Control Station; but for reasons of subsystem maturity, component availability/compatibility, and for working group partitioning efficiency, these AIP, SE, and Teleop functions have been allocated outside the OCS.

OCS's OPERATOR REQUIREMENTS

The human Operator uses the OCS to interface with the system. The Operator manages system configuration, transmits system information and receives feedback from the System. The OCS provides capability for the Operator to coordinate and monitor all other subsystems, permits the Operator to direct/supervise/execute robotic and teleoperation control. In the JPL Telerobot Demonstrator System, OCS is designed for two operators, the Main Operator and the Auxiliary Operator (also known as the Test Conductor). The Main Operator has the capability to execute all functions regardless of the absence/presence of the Auxiliary Operator. The Operator's functions include the following:

- System management functions: system startup/shutdown, setup, software configuration, other monitoring/diagnostics functions;
- System operation functions: mode transitions, setting system parameters, system calibration, video switching, emergency halt (and other modes of halting), object data

manipulation;

- Teleoperation functions: hand motion for input to hand controllers, setting subsystem parameters, establishing telepresence via visual, kinesthetic and proprioceptive feedback;
- Robotic control functions: instantiate, monitor, supervise, direct, confirm and give permission to proceed all actions generated under autonomous planning;
- Trading and sharing teleoperated and robotic controls;
- Initiating and executing data logging functions for off-line analysis and system performance evaluation.

OCS HARDWARE REQUIREMENT

The OCS hardware is a station, in a "controlled" room environment where lighting, sound and sight are controllable, and houses the Operator and Test Conductor. The station is equipped with multiple monitors for video and graphics displays and mixing. Audio and voice input/output systems are provided for operator command inputs in addition to keyboard inputs. Mechanical input devices for teleoperation and shared robotic/teleoperation are provided. Multiple processors and computer networking is provided for OCS functions, planning functions, and system management functions. The OCS is designed with due consideration in anthropometric and ergonomic constraints. Specifically, the requirements are:

1. Be configured with workstations for an Operator and a Test Conductor;
2. The main workstation to have all necessary control inputs and feedback;
3. At the main workstation, to provide the following devices for display, record and transmit data:
 - (a) mono, stereo video monitors for camera and buffer images
 - (b) video switcher and mixer(s)
 - (c) indicator lights, aural signals
 - (d) synthesized voice
 - (e) video recorder(s), audio mixer
 - (f) high resolution bit-mapped video display monitors;
4. At the main workstation, to provide the following devices to accept Operator requests and commands:
 - (a) keyboard(s), function switches
 - (b) one left and one right bilateral force reflecting 6-dof hand controllers; the mechanisms and electronics
 - (c) voice input/recognizer

(d) mouse(s) and joystick(s);

5. At the auxiliary workstation (of the Test Conductor), to provide:

- (a) all data/views available at the main workstation except the stereo view
- (b) peripheral data collection hardware for off-line data analysis
- (c) keyboard function switches independent of main workstation;

6. To provide the following physical and electrical support:

- (a) regulated electrical power
- (b) VCR recorder
- (c) terminal/keyboard interface to the AIP (for autonomous and shared control instantiation, for object designation process)
- (d) terminal/keyboard interface to the SE (for executing system management functions);

7. To provide display system to permit:

- (a) terminal emulation through windowing to allow the Operator to access each subsystem
- (b) switching of video images onto designated monitors;

8. At both the main and auxiliary workstations, to provide a "Panic Button" for emergency halt and other mode(s) of halt;

9. To provide networking capability to all subsystems;

10. To provide electronics, hardware, switches to permit teleoperation mode changes, indexing, triggering for robot end-effector opening and closing, force reflection, and other control processing for the hand controllers;

11. In support of Operator-assisted object designation and verification, to provide:

- (a) mouse/keyboard/voice input for selection of objects
- (b) graphics generation, mixing capability to overlay wire frame models of objects onto video images;

12. To provide the OCS computer(s) to perform OCS internal and interface functions.

Figures 2 and 3 show the control station configuration of the JPL Telerobot Demonstrator System. Figure 4 show the functional block diagram of the OCS, illustrating the interface with all the subsystems.

OCS SOFTWARE REQUIREMENTS

Software required in OCS includes the processing of OCS input/output data; interface software with other subsystems; the system management software (currently allocated to SE); and the system mode switching/

supervision, object-data-manipulation, man-machine-interface (currently allocated to AIP); and hand controller teleoperation software (currently allocated to MCM/Teleop). Specific software requirements are to provide:

1. Command interpreter to process Operator generated commands via the keyboard, mouse, and voice; hence, to parse, translate and generate inter- and intra-OCS commands;

2. Message processor for translating, generating and displaying messages on OCS monitor; for messages initiated from within or outside OCS;

3. Status monitoring process, to monitor OCS health and parameter-setting status; in addition to the overall system and subsystem status monitoring process performed by the SE;

4. RS-232 controllers for graphics overlay/mixer controller, voice display controller, and video switch controller;

5. Gateway computer interface via an ethernet network;

6. NIP (a custom Network Interface Package) gateway interface software for processing NIP transactions [3].

7. Object designation/definition software to create wire-frame models, overlay on camera images, manipulate using mouse cursors, perform best-fit, and update/augment data bases; to interface with the AIP for model/data;

8. Interface software between the primary and auxiliary OCS workstations;

9. Terminal emulation to all subsystems via multiple windows on OCS monitor;

10. Pull-down menus for system and subsystem commands;

11. Continuous speech voice recognition and multi-voice (gender/person) speech synthesis for direct input and feedback;

12. Graphics generation and overlay software for object designation/verification process;

13. Graphics and simulation software for special displays including predictive displays, scenario simulation, sequence playback displays;

14. Software to accept keyboard/mouse/symbolic/graphics input, instantiate processes, specify goals and task sequences, as specified by Operator;

15. Software to allow operator to display/supervise/update/modify/cancel task sequences and parameters;

16. Software and displays for smooth transition between system modes, including teleoperation mode to and from robotic mode;

17. Data base maintenance, management, and creation of world models, object location,

camera models, and calibration settings;

18. Support software for interpreting and transmitting Operator generated (voice or keyboard) manipulator control commands; maintain command context and monitor status of commands;

19. System configuration, startup/shutdown/halt, statusing, health monitoring, data traffic monitoring of all subsystems;

20. Control processing and data acquisition software to process Teleoperation hand controller joint data, special switches, triggers, force feedback, coordinate transformation and communication;

21. Graphics processing/generation of Teleoperation related displays including force/torque data and predictive data.

Requirements #1-13 are currently allocated to the OCS of the JPL Telerobot Demonstrator System, while #14-18 are allocated to the AIP, #19 to SE, and #20-21 to the MCM/Teleop for reasons mentioned in an earlier Section of this paper.

OCS in the NASREM ARCHITECTURE

The NASA/NBS Standard Reference Model (NASREM) [4] for telerobot systems functionally partitions the telerobot process into six levels, where each level has its sensory processing, world modelling and task decomposition subprocesses. Levels interconnect with the level below and the level above it. All levels connect with a 'global data base' and an 'operator interface'.

The OCS described in this paper, i.e. the physical station and its functions, map right into the 'operator interface' block in the NASREM architecture. A few of the functions, particularly those now allocated to AIP, SE, and MCM, but considered above to be OCS functions, overflow out of the NASREM 'operator interface' block. Figure 5 (excerpted from [5]) shows the mapping of the JPL Telerobot Demonstrator System into the NASREM.

FUTURE RESEARCH and DEVELOPMENT

Upon its completion, installation and integration in mid 1989 with the rest of the System, the OCS will serve as the focal point of the Telerobot Demonstration System. Real hands-on operational flow analysis and workload analysis could actually be conducted, to evaluate the

effectiveness of the OCS design.

More research and development items, improvements on point-designs, alterations of physical layout, addition of vocabulary, etc. will undoubtedly surface when more experience is gained from OCS experiments. Other already foreseen technology development items include: interactive model/data base building; the use of CAD-type data base techniques for object trajectories planning/verification; faster and better algorithms in object designation process, including hidden line removal, incorporation of perspective, cues etc.; smoother and more unified operator-machine interface; more powerful display/graphics systems. As more powerful computers become available, and as understanding of a telerobot system matures, the state-of-the-art OCS technology will evolve.

ACKNOWLEDGEMENT

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

- [1] P. Schenker, "Telerobot Program Objectives and Technology Outreach," Proceedings of the Workshop on Space Telerobotics, Pasadena, CA, Jan. 1987 (also JPL Publication 87-13, Vol. 1, pp. 3-17, July 1987).
- [2] J. Matijevic, et.al., "Functional Requirements for the Telerobotic Testbed," Jet Propulsion Laboratory Document, #JPL D-3693, May 1988.
- [3] R. Cain, et.al., "Network Interface Package (NIP) User's Guide," SRI International, Contract #957908 Report to JPL, also SRI Project #3528, Nov. 1987.
- [4] J. Albus, H. McCain, R. Lumia, "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," National Bureau of Standards, NBS Technical Note 1235, Jun. 1987.
- [5] W. Keks and J. Matijevic, "NASREM and JPL Telerobotic Testbed," Jet Propulsion Laboratory Internal Publication 347-88-319, May 1988.



Figure 1. The JPL Telerobot Laboratory
with a Generic Operator Control Station

FRONT VIEW

ORIGINAL PAGE IS
OF POOR QUALITY

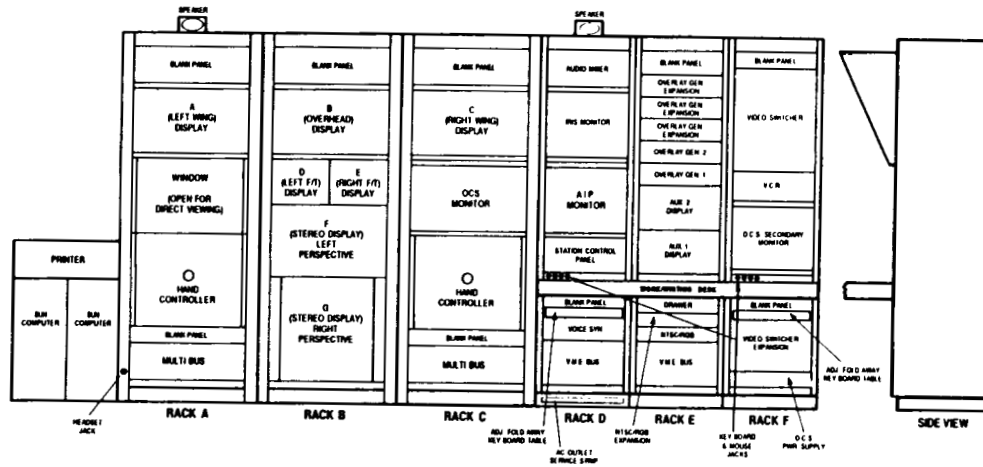


Figure 2. The OCS - Elevation View

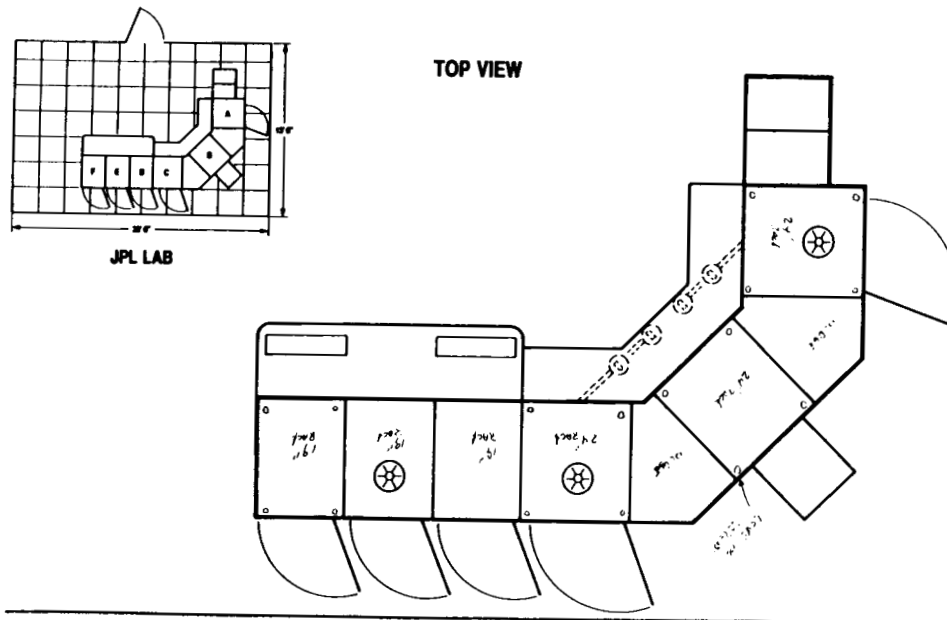


Figure 3. The OCS - Plan View

ORIGINAL PAGE IS
OF POOR QUALITY

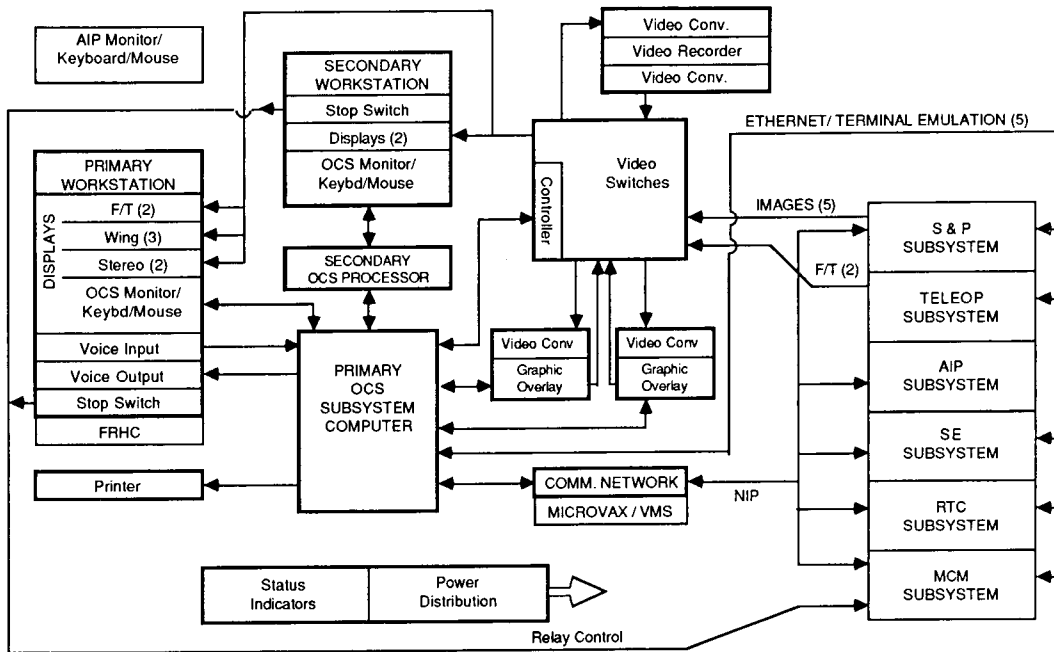


Figure 4. TELEROBOT OPERATOR CONTROL STATION
FUNCTIONAL BLOCK DIAGRAM

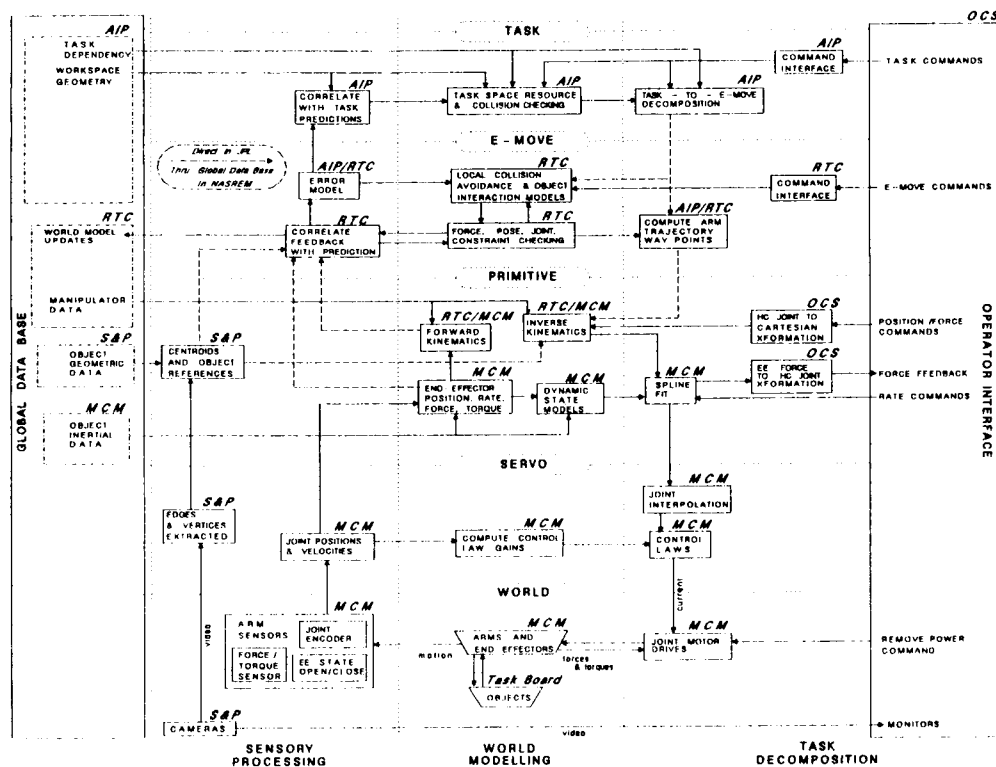


Figure 5. JPL Telerobot System Mapped into NASREM

TIME-DELAYED OPERATION OF A TELEROBOT VIA GEOSYNCHRONOUS RELAY

Brian H. Wilcox
 Supervisor- Robotics Systems Research,
 Implementation, and Integration Group
 Jet Propulsion Laboratory
 California Institute of Technology
 4800 Oak Grove Drive
 Pasadena, California 91109
 (818) 354-4625

ABSTRACT

Operation of a telerobot is compromised if a time delay more than a few hundred milliseconds exists between operator and remote manipulator. However, the most economically attractive way to perform telerobotic functions such as assembly, maintenance, and repair in Earth orbit is via geosynchronous relay satellites to a ground-based operator. This induces loop delays from one-half to two seconds, depending on how many relays are involved. Such large delays makes direct master-slave, force-reflecting teleoperated systems infeasible. Research at JPL on a useful telerobot that operates with such time delays is described.

INTRODUCTION

It has long been recognized that the performance of a master-slave teleoperator is seriously degraded if the closed-loop latency between input at the master, action at the slave, and perception of the result back at the master is more than a few hundred milliseconds [1]. It is generally felt that loop delays of a few milliseconds are acceptable, that delays of a few hundred milliseconds can be overcome with extensive training (untrained operators are unstable in their attempts to guide the slave with such delays), and that delays of 500 milliseconds or more cause operators to revert to a 'move and wait' strategy for manipulator control. The move-and-wait strategy gives very low performance compared to normal teleoperation, and is still prone to errors and damage due to undesired contact between the slave manipulator and its environment.

The space tasks of greatest interest to NASA and the Air Force are those in low Earth orbit, and of course the most economical place for the operator is on the ground. Continuous communication between these two sites is most easily accomplished via one or more geosynchronous communication satellites, at an altitude of 22,300 miles above the equator. The time delay at the speed-of-light from Earth or low orbit to geosynchronous orbit is about 120 milliseconds. Thus the most straightforward and economical arrangement of a teleoperator for space tasks has a round-trip latency of 1/2 to 1 second (actually 480 milliseconds if only one geosynchronous relay is used, and double that if line-of-sight considerations cause two relays to be needed). In fact, the delay can be as much as two seconds if the ground operator is not located at one of a few Earth-station points such as White Sands or Palo Alto, since a second satellite link would be used from the operator's location (e.g. Houston) to the prime Earth-station site. Thus the minimum latency of a geosynchronous relay will cause serious degradation of the performance of a conventional teleoperator system.

(Operation by astronauts is of course possible without significant time delay, but astronaut time is very expensive, and many U.S. space assets are in orbits that cannot be reached by the Space Shuttle.)

A TIME-DELAYED TELEROBOT

Supervisory Control was proposed a decade ago to deal with this problem [2]. This paper describes a particular implementation approach for supervisory control being studied at JPL as part of the Telerobotics Research program sponsored by the NASA Office of Aeronautics and Space Technology. The architecture of this telerobotic system is as shown in figure 1.[3]

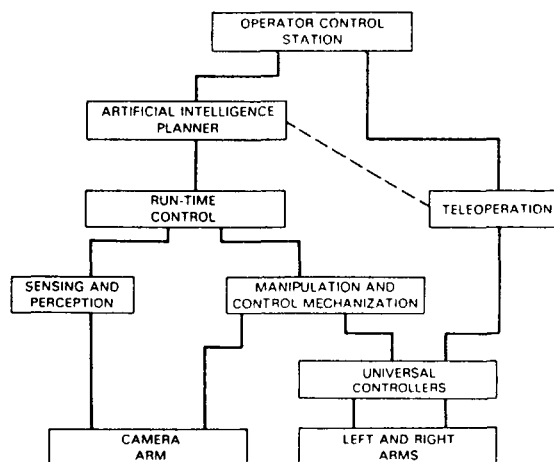


Figure 1. Telerobot Subsystem Architecture

The subsystems of this architecture which interact with the environment are the Sensing and Perception subsystem and the Manipulator Control Mechanization Subsystem (S&P and MCM) [4][5]. The actions of these subsystems are coordinated by the Run-Time Control subsystem (RTC) [6]. The RTC contains the task data base, performs fine-motion planning, requests verification of the locations and orientations of objects by the S&P subsystem, and commands trajectory via points and force/torque task frames for execution by the MCM subsystem. The Artificial Intelligence Planner (AIP) performs gross motion planning and automatic task sequencing, and gives the operator a 'window' into the more autonomous operations of the telerobot so that planned actions may be reviewed and modified [7][8]. The teleoperation channel allows force-reflecting hand controllers

(FRHC's) to directly control the manipulators, with forces and torques detected at the wrists of the manipulators and used to backdrive the FRHC's.

It is of great interest to determine which subsystems must be resident at the remote site, since of course the flight qualification of high-performance computational elements is very demanding. Ground-based computers can be extremely powerful compared to space-qualified computers, and of course weight and power consumption are practically negligible considerations on Earth. Thus an important objective of research with the telerobot testbed is to determine the minimal flight elements of a useful telerobot, given the time delay of the geosynchronous relay link.

Physically, all the subsystems of the telerobot testbed are connected via ethernet (except the teleoperation channel, which uses a high-speed parallel channel to the Universal Controllers [9]), allowing general communication protocols and easy reconfiguration. This further allows the simulation of different assignments of subsystems to the operator site or the remote site, by inclusion of a time-delay simulation in the forwarding of the ethernet packets. It has long been recognized that a simulation of system loop delay need only put the entire delay in one leg or the other of the communication system, and that it is not necessary to split the delay between uplink and downlink, as it is in a real operational system [1]. Since the downlink from the telerobot will include several channels of video, and since this will ultimately be high-quality color video (e.g. red, green, and blue transmitted separately), the bandwidth of the downlink can easily approach 1 Gigabit per second. (Of course, operational considerations may force the use of slow-scan, image compression, and other techniques to reduce this, but these expediencies are outside the scope of the current testbed focus of basic telerobotic research.) The uplink, on the other hand, will have a few tens of ethernet packets per second, at a few hundred bits each, plus data from the two FRHC's consisting of the twelve encoder values sampled at a kilohertz (our current rate, which gives very good results), for a total of a hundred kilobits per second. Thus delay simulation is most easily accomplished by buffering the uplink. We are currently designing a time-delay simulation element of the architecture which essentially 'gateways' messages on the ethernet between subsystems declared to be at the operator site and those declared to be at the remote site, as well as buffering the FRHC data. The 'downlink' ethernet packets, arm position and reflected force data, as well as video and other sensor data, are not delayed.

The minimal initial flight segment of a telerobot could consist (in the subsystem breakdown described above) of all or parts of the MCM (Manipulator Control Mechanization) and S&P (Sensing and Perception) subsystems, since these are the subsystems that directly interact with the environment. The MCM subsystem is configured as two distinct elements, the real-time part and the non-real-time part. Clearly the real-time part, which performs position and force servoing of the manipulators, needs to fly with the telerobot. If very slow manipulation speeds are acceptable (and they may be dictated by safety concerns anyway), one or a few MIPS (Million Instructions Per Second) and 300 KFLOPS (Floating Point Operations Per Second) will probably be adequate (a microVAX-II at about 0.75 MIPS and 100 KFLOPS has been used in our recent demonstrations of telerobotic capability, which included grappling and docking with a satellite mock-up, door opening, crank turning, etc.). If flexible arms are used or faster manipulations (10's of cm/sec) are needed (requiring dynamics computations) then perhaps 10-20 MIPS and 1-10 MFLOPS will be needed for arm control.

Of the Sensing and Perception subsystem, initially one could fly just the video cameras. Everything that is currently done in the testbed S&P subsystem could be done on the ground if necessary, including the real-time edge detection used for grappling the satellite. This results from the fact that angular momentum is conserved with such high accuracy in orbit that the computation of the position, velocity, orientation, and angular-velocity on the ground can be very accurately projected a few seconds ahead in time. This requires accurate knowledge of the inertia tensor of the satellite, which can be computed from the CAD data base used in manufacture or derived from a few extra minutes of tracking the satellite. Once the satellite is grappled and docked, all current S&P computations are verifications of static object positions, and so the time delay will not seriously affect overall system performance. Likewise the Run Time Controller and Artificial Intelligence Planner can be located on the ground, as the small amount and low frequency of communication with the flight segment of the telerobot, and its non-time-critical nature is exemplified by our use of the ethernet for communication among these subsystems. Also, the non-real-time portion of the MCM subsystem (the trajectory generator) does not really need to fly, since the form of uplink data can be intermediate trajectory points or spline-fit parameters for free-space motion, task frame definitions for compliant motion (e.g. those axes in cartesian space that should be position controlled and those that should be force or torque controlled), the expected force/torque envelopes, and error conditions for abort.

If it becomes necessary to fly more of the S&P subsystem (for example, to assemble a large trusswork that is constantly vibrating), one possible approach being researched at JPL is for a multi-resolution pyramid image processor to be flight qualified. This processor, which filters and subsamples images to form a pyramid of low-pass and band-pass images (e.g. 512x512, 256x256, 128x128, etc.), takes about 200 MIPS to continuously compute the pyramid on a 30 Hz image stream at 512x512 resolution. This pyramid machine can be built using 3x3 convolver chips developed by JPL for a machine vision research tool, and which have been designed to be readily flight-qualified. This would permit real-time tracking, object acquisition, and reflex actions.

Other partitions of the flight and ground systems are possible. However, it does not seem advisable to partition S&P except at the camera outputs or after the feature extraction step due to the huge amount of parallel data being processed during feature extraction (some 10 parallel image paths at 10 Mpixels/sec for a total of about one Gbit/sec), which would swamp the available communications channels. Once one has gone to all the trouble of doing the feature extraction (~1 billion 12-bit fixed point operations per second), we might as well do all the rest of the S&P computations at the remote site (10-30 MIPS and 3-30 MFLOPS). RTC may take 10-100 MIPS and 10-50 MFLOPS as well, and is a good candidate for a flight hypercube or other concurrent architecture. As mentioned before, MCM could take up to 10-20 MIPS and 1-10 MFLOPS if manipulation speeds call for dynamics computations or if somewhat flexible arms are used.

The ground segment of the minimal flight telerobot need not be very complex to be highly effective. The recent demonstrations of telerobotic capability at JPL allowed the operator to use a wire-frame overlay on the video image returned from the remote site to designate objects. More recently, RCA, under contract to JPL to create an integrated Operator Control Station for the telerobot testbed, has demonstrated the use of wire-frames for 'analogic' designation in conjunction with voice input and output for 'symbolic' designation. One can certainly imagine an operator using the

two six-degree-of-freedom hand controllers to control the position and orientation of wire-frame overlays while using voice control to call up named objects or give new names to object models being created. The operator, by matching the position of the wire-frames on the unprocessed video from the telerobot site in a stereo display, will be able to achieve millimeter precision in 'telling' the control system where objects are located (machine vision techniques can refine this further if needed). The operator can then invoke 'skills' such as bolt removal, handle grasping, or other operations involving contact with the environment using a combination of analogic 'pointing' and symbolic voice inputs. A repertoire of a few dozen 'skills' will allow an operator to conduct a complex task without any significant advance preparation or extensive data base, with or without a time delay in the control loop. If a complete task data base is available, then a 'virtual force field' can be created around objects in the environment and used to backdrive the FRHC's to avoid contact while teleoperation is underway. The only physical contact permitted by the system would be when control is 'traded' to the autonomous system for execution of a skill, or when 'shared' control is invoked, where the autonomous system controls all forces while the human directs motion in unconstrained directions. Demonstration of this shared and traded control methodology is a principal objective for the telerobot testbed in the coming year.

CONCLUSIONS

A near-term telerobot flight segment needs only 1) video cameras and 2) the inner core of the servo-control system, able to maintain stable control over the arms while in free space motion, to decelerate smoothly near the task, to move very slowly in a guarded move to the instant of contact, and to switch to force control for executing the appropriate compliant frame definition for the task at hand. A processor with as little as 2 MIPS and 300 KFLOPS should be adequate for this minimal time-delayed telerobot. The downlink would be video at a few hundred Mbits per second, and the uplink would be trajectory spline parameters, compliant frame definitions, and error condition predicates at ~100 Kbits per second.

In the longer term it may become advantageous to fly the entire MCM, S&P, RTC, and part of the AIP subsystems (some of the Artificial Intelligence Planner will always be located at the operator control station for replanning and to give advice). This, however, will only improve the performance by an order-of-magnitude or so over the minimal system described above (which has some four to five orders-of-magnitude economic advantage over the use of astronauts for these tasks). The computational load of the flight subsystems will jump from a few MIPS and some 0.3 MFLOPS to perhaps 30-100 MIPS and 10-50 MFLOPS of general-purpose computer and 1000's of MIPS of image processor. Thus great advances in flight-qualified computation are needed to evolve much beyond the minimal time-delayed telerobot. Yet the minimal time-delayed telerobot, which can be configured from the imminently flight-qualified NS32016 or 80386 processor families, offers huge economic benefits for orbital assembly, maintenance, and repair tasks.

ACKNOWLEDGMENT

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

1. T.B. Sheridan and W.R. Ferrell, 'Remote Manipulative Control with Transmission Delay,' *IEEE Transactions on Human Factors in Electronics*, p 25-29 (September 1963).
2. T.B. Sheridan and W.R. Ferrell, *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*, MIT Press, Cambridge MA, 1974.
3. W. F. Zimmerman and J. R. Matijevic, 'System Engineering Techniques for Establishing Balanced Design and Performance Guidelines for the Advanced Telerobotic Testbed,' *Proceedings of the Workshop on Space Telerobotics*, JPL publication 87-13, vol 1, p 67-74.
4. B. Wilcox, D.B. Gennery, B. Bon, and T. Litwin, 'The Sensing and Perception Subsystem of the NASA Research Telerobot,' *ibid*, Vol 2, p 3-8.
5. S. Hayati and B. Wilcox, 'Manipulator Control and Mechanization: A Telerobot Subsystem,' *ibid*, Vol 2, p 219-234.
6. J. Balaram, A. Lokshin, K. Kreutz, and J. Beahan, 'A Run-Time Control Architecture for the JPL Telerobot,' *ibid*, Vol 1, p 211-222.
7. C. Collins and M. Rokey, 'Planning for the JPL telerobotics project,' *Proc. 4th Annual Artificial Intelligence and Advanced Technology Conference*, p 429-437, May 1988.
8. D. Mittman, 'Audrey: An Interactive Simulation and Spatial Planning Environment for the NASA Telerobot System,' *ibid*, p 421-428.
9. A. K. Bejczy and Z. Szakaly, 'Universal Computer Control System (UCCS) for Space Telerobots', *IEEE International Conference on Robotics and Automation*, March-April 1987, vol 1, p318-325.

TBA

Joe Wong
AF Space Division

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

A Representational Framework and User-Interface for an Image Understanding Workstation

Joyce D. Schroeder
Speech and Image Understanding Laboratory
Computer Science Center
Texas Instruments
P.O. Box 655474, M.S. 238, Dallas, TX 75265

Abstract

Problems in image understanding involve a wide variety of data (e.g., image arrays, edge maps, 3-D shape models) and processes or algorithms (e.g., convolution, feature extraction, rendering). This paper describes the user-interface and underlying structure of an Image Understanding Workstation designed to support multiple levels and types of representations for both data and processes.

The Image Understanding Workstation consists of two parts: the Image Understanding (IU) Framework, and the user-interface. The IU Framework is the set of data and process representations. It includes multiple levels of representation for data such as images (2-D), sketches (2-D), surfaces (2-1/2 D), and models (3-D). The representation scheme for processes characterizes their inputs, outputs, and parameters. Data and processes may reside on different classes of machines.

The user-interface to the IU Workstation gives the user convenient access for creating, manipulating, transforming, and displaying image data. The user-interface follows the structure of the IU Framework and gives the user control over multiple types of data and processes. Both the IU Framework and user-interface are implemented on a LISP machine.

By developing a fundamental set of representations and a consistent user-interface, the IU Workstation provides a rich environment for algorithm developers and system engineers to implement and study applications in image understanding.

and more abstract, symbolic representations such as lists of edges. Appropriate representations must exist in an IU system to accommodate these different levels of information. The characteristic computations in IU work are also diverse, ranging from numerically intensive to symbolic. The goal of the Image Understanding Workstation described in this paper is to provide an integrated and highly interactive software environment to support such diverse IU activity.

In general, IU software environments are evolving into integrated systems of considerable computational and representational power [4]. Some recent systems illustrating these trends are: Visions [2], Powervision [5], and SuperSketch [6]. As described in the review article by McConnell and Lawton [4], the basic components of general IU environments are: representations, programming constructs, system-specific databases, and user-interfaces.

Several current IU environments are built on top of object-oriented programming environments that readily support data and process abstraction. Key attributes of object-oriented environments are: abstract type definition, combination and inheritance mechanisms, and access to diverse objects through a uniform set of messages [4].

The Image Understanding Workstation described in this paper consists of two parts: the Image Understanding (IU) Framework, and the user-interface.

IU FRAMEWORK

The IU Framework consists of the fundamental set of data and process representations in the IU Workstation. As shown in Figure 1, the IU Framework includes representations for IU data and a core set of IU algorithms. The IU Framework is intended to be a foundation for other program modules such as control strategies and application-specific environments.

INTRODUCTION

Problems in image understanding involve a wide variety of data (e.g., image arrays, edge maps, 3-D shape models) and processes or algorithms (e.g., convolution, feature extraction, rendering). A critical issue in image understanding (IU) is the representation of data. Data in IU work includes initial sensor-derived numeric arrays

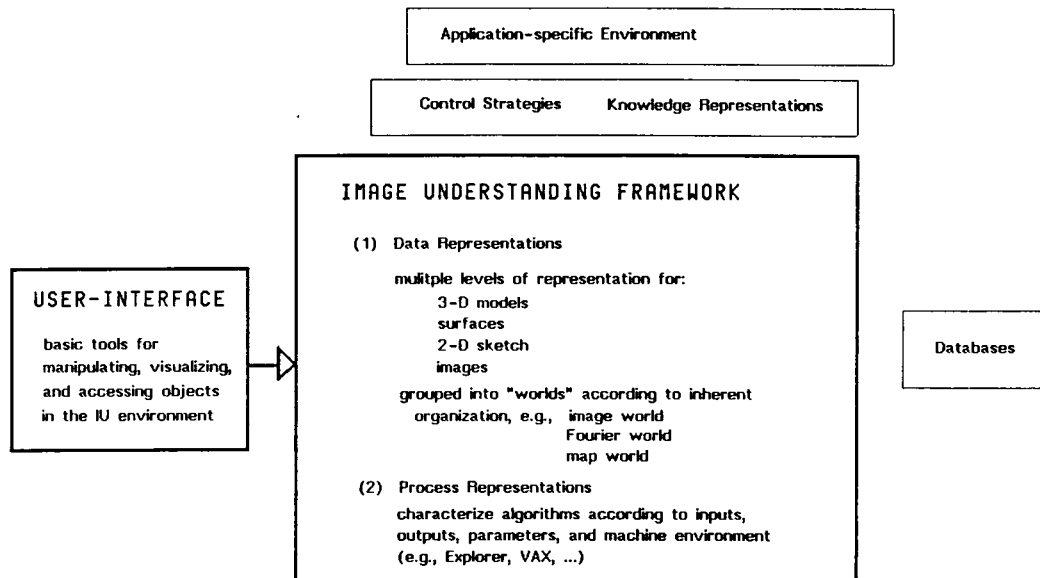


Figure 1.

(i) Data Representations

Problems in image understanding involve data at different levels of abstraction, ranging from numeric arrays created by a sensor, to extracted feature objects such as curve lists. The goal is to provide a range of data representations appropriate for each of these types of information.

Data objects share several kinds of information such as: data type, data, name, domain and range dimensionality, and axis information which completely describe the underlying data. In addition, data objects have an associated display-options attribute which specifies the default way to display the object in the IU Workstation.

Figure 2 illustrates two different types of data objects in the IU Workstation: the bridge image in the left window is a 2D-ARRAY, and the set of extracted contours in the right window is a 2D-INSTANCE-LIST. The underlying data for the extracted contours is a list of contour line objects (not a one-bit array). Each contour line object in the list is a data object characterized by: number, type, length, elevation, list of points, thickness, and color. Further data abstraction to lines of a minimum length is illustrated in Figure 3.

Data objects in the IU Workstation can be created using data from either LISP arrays, VAX files, or Odyssey [1] format files. Appropriate defaults are provided for axis information and display options.

Data objects in the IU Workstation are assigned to

"worlds" according to the inherent organization of the data in the object. For example, an Image World has objects whose data is addressed by rows and columns in an array. Map World data is accessed by latitude, longitude, and elevation; Hough World data is accessed by radius and angle. The use of "worlds" classifies data objects by type, allowing processes and methods selective access to particular groups of data objects.

The relationship between data objects and process objects is shown in Figure 4.

(ii) Process Representations

Processes or algorithms are also implemented as objects in the IU Workstation. A process is characterized by its input data-object(s) and world, output data object(s) and world, parameters, and class of machine. Processes are invoked via an evaluate method that automatically saves a copy of the information about the process being executed: its inputs, outputs, and arguments.

Processes in the IU Workstation are not limited to the LISP environment. A Remote Processing Object has been implemented to forward commands and monitor responses from remote hosts. Currently, the Remote Processing Object uses the Telnet window protocol to communicate with VAX systems. This integrated approach allows numerically intensive operations to be sent to an appropriate machine for computation. The IU Workstation provides the entire interface for the remote process, collecting pathnames and arguments, then sending the instruction to the remote host. Remote computations can be monitored from the IU Workstation and output files

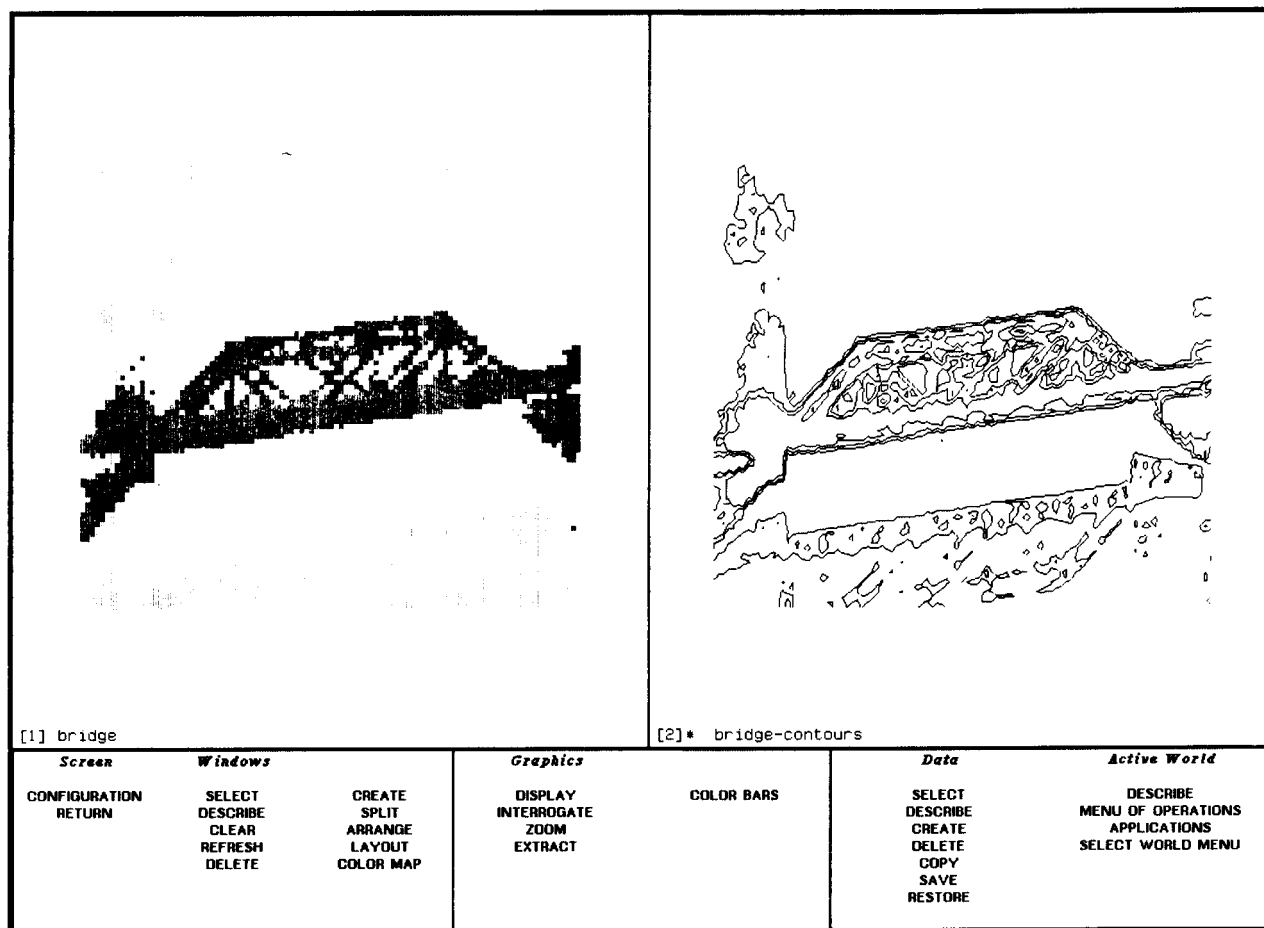


Figure 2.

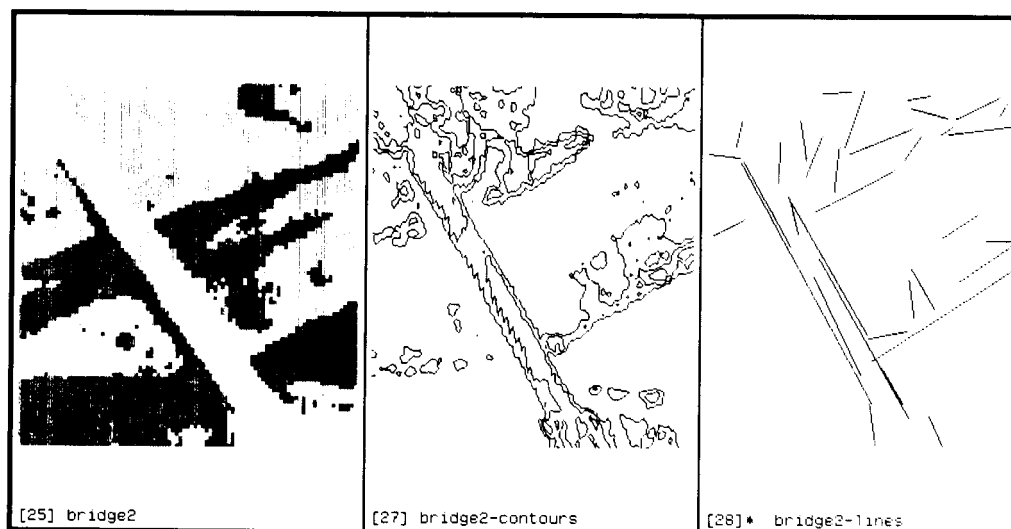


Figure 3.

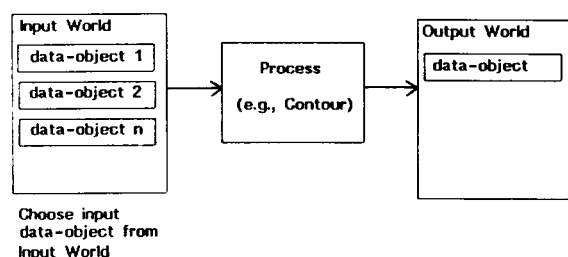


Figure 4.

can be automatically set up as data objects in the IU Workstation.

Figure 5 shows the results of an algorithm executed through the Remote Processing Object. The input image is shown in the upper left window. A VAX-based FORTRAN scene segmentation program was used to create the output segmented images shown in the upper right and lower left windows. A pop-up menu showing the parameters used in the Scene Segmentation process is also displayed.

Note that processes are also implemented as objects in the IU Workstation. As such, they can be treated as "data" in the sense that higher level control strategies or knowledge representations can be constructed to manipulate or reason about process objects. (See Figure 1.)

USER-INTERFACE

The IU Workstation user-interface is the set of tools for accessing, manipulating, and visualizing objects in the IU environment. The user-interface is window-oriented and gives the user a consistent set of commands for managing a complex set of image data and processes which reside either on the LISP machine or a remote machine.

The goal of the IU Workstation user-interface is to give the user simple and rapid communication with the IU environment through mouse-button selectable commands, menus with appropriate default settings, and pop-up description windows for all objects in the environment: data objects, process objects, and windows and associated displays.

The screen of the IU Workstation is comprised of several types of windows: command menus, display windows, and a Lisp typein window. The screen can be reconfigured with fewer command menus in order to maximize display space. The three bottom-level menus shown in Figure 5 contain the basic commands for window reconfiguration, graphics, and data manipulation.

IU Workstation commands are selectable by mouse, assigned keystroke, or typein, and extensive help and documentation are a part of each command definition. In general, commands in the IU Workstation are mouse-button specific. That is, left mouse clicks on a command give default behavior; middle or right mouse clicks give menus. Default behaviors (e.g., default parameters for displays, or default input and output worlds for processes) allow the user to rapidly specify simple tasks.

Pop-up command menus are used in the IU Workstation to hold less frequently used commands. Screen space is saved by grouping these commands into a pop-up menu that is accessed by an assigned keystroke. For example, commands related to the Remote-Processing-Object are handled in this manner.

The commands in the IU Workstation provide a common interface (or set of messages) to the different data objects in the IU environment. For example, the DISPLAY command uses appropriate display code according to whether or not the selected data object is a 2D-array, 2D-instance-list, or other data type. Similarly, a DESCRIBE command is applicable to all objects in the IU environment.

Another key feature of the IU Workstation user interface is the functionality included in the display windows. Display windows have an associated virtual window which describes the mapping between displayed data values and window coordinates. Thus, display windows support operations for zooming, copying, interrogating, and extracting portions of the displayed data using the mouse. The IU Workstation display windows maintain information about the data being displayed, and also how it is displayed (i.e., magnification, color map, annotation, etc.) Display capability exists for both color and black and white monitors. The IU Workstation is implemented on the Texas Instruments Explorer II Lisp Machine.

SUMMARY

The Image Understanding Workstation described in this paper is implemented in an object-oriented programming environment and includes multiple levels of representations for IU data. Processes or algorithms are also characterized and computations on different classes of machines are supported.

The user-interface to the IU Workstation is composed of basic tools for visualizing, manipulating, and accessing objects in the IU environment. The user-interface is highly interactive, window-oriented, and gives the user a consistent set of commands for managing the complexity inherent in IU environments.

ORIGINAL PAGE IS
OF POOR QUALITY

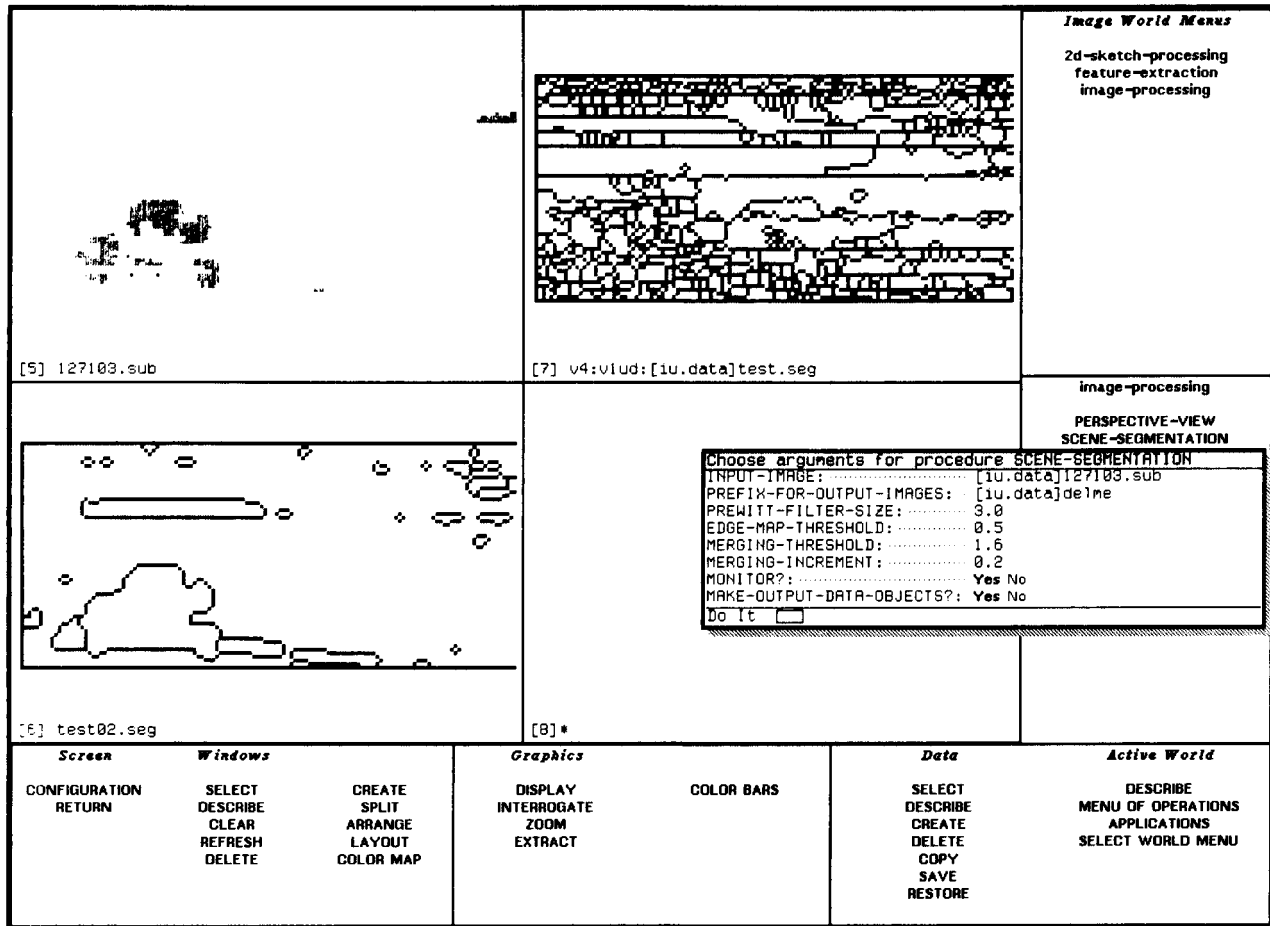


Figure 5.

By developing a fundamental set of representations and a consistent user-interface, the IU Workstation provides a rich environment for algorithm developers and system engineers to implement and study applications in image understanding.

ACKNOWLEDGMENTS

The Image Understanding Workstation has been jointly developed by Bruce Flinchbaugh, Joyce Schroeder, Marion Lineberry, and Bob Gove. All are members of the Speech and Image Understanding Laboratory, in the Computer Science Center, Texas Instruments.

REFERENCES

1. Gove, R.J., "Integration of Symbolic and Multiple Digital Signal Processors with the Explorer/Odyssey for Image Processing and Understanding Applications", *Proceedings of the IEEE International Symposium on Circuits and Systems*, Philadelphia, Pennsylvania, May, 1987, pp.968-971.
2. Hanson, A. and Riseman, E., The VISIONS Image Understanding System- 1986, In Chris Brown, editor, *Advances In Computer Vision*, Erlbaum Press, 1987.
3. Kopec, G.E., "The Integrated Signal Processing System ISP", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 4, August, 1984, pp. 842-851.
4. McConnell, C.C., and Lawton, D.T., "IU Software Environments", *Proceedings: Image Understanding Workshop*, Vol. II, Cambridge, Massachusetts, April, 1988, pp.666-677.

5. McConnell, C.C., Nelson, P.C., and Lawton, D.T., "Constructs for Cooperative Image Understanding Environments", *Proceedings: Image Understanding Workshop*, Vol. II, Los Angeles, California, February, 1987, pp. 497-506.
6. Pentland, A.P., "Perceptual Organization and the Representation of Natural Form", *Artificial Intelligence* 28, 1986, pp. 293-331.

Machine Vision for Space Telerobotics and Planetary Rovers

Brian H. Wilcox
Supervisor- Robotics Systems Research,
Implementation, and Integration Group
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109
(818) 354-4625

ABSTRACT

Machine vision allows a non-contact means of determining the three-dimensional shape of objects in the environment, enabling the control of contact forces when manipulation by a telerobot or traversal by a vehicle is desired. Telerobotic manipulation in Earth orbit requires a system that can recognize known objects in spite of harsh lighting conditions and highly specular or absorptive surfaces. Planetary surface traversal requires a system that can recognize the surface shape and properties of an unknown and arbitrary terrain. Research at JPL on these two rather disparate types of vision systems is described.

INTRODUCTION

The JPL Robotics Laboratory has been conducting sensing and perception research since the mid 1970's, when a task was undertaken to develop a breadboard Mars rover which could navigate autonomously over unknown terrain. At that time, and continuing to the present, the principal sensor modality addressed was machine vision. This arises from the fact that it is essential, both in planetary rover and orbital tasks, to sense the environment prior to actual physical contact so that contact forces can be controlled. The available non-contact sensing techniques are limited to those based on electromagnetic radiation and those based on sound. Obviously sound is not useful in vacuum and of limited use in extremely rarified atmospheres. Electromagnetic sensing can be of an active type, emitting radiation and sensing the reflection, or passive, relying on ambient radiation. Active sensing systems can give direct information such as object range, but often consume excessive power and involve mechanical scanning devices which are potentially unreliable. Thus passive electromagnetic sensing is an attractive means of accomplishing the non-contact sensing function. The only wavelengths for which large amounts of ambient radiation exist in space are those emitted by the Sun, i.e. visible light and near IR. Sensors for these wavelengths are readily available with very good spatial and temporal resolution and accuracy in the form of solid-state video cameras. This has the further advantage that the human operator can easily comprehend the raw data from these sensors using a video display.

More recently, machine vision research at JPL has been extended to applications for near-Earth orbit. A useful space telerobot for on-orbit assembly, maintenance, and repair tasks must have a sensing and perception subsystem which can provide the locations, orientations, and velocities of all relevant objects in the work environment. Examples of the potential uses of such technology are robotic systems for capturing satellites which have arbitrary and unknown motion, and robotic systems for construction in space.

VISION FOR SPACE TELEROBOTICS

The sensing and perception subsystem of the Telerobot Testbed at JPL is designed to acquire and track objects moving and rotating in space, and to verify the locations of fasteners, handles, and other objects to be contacted or avoided during the space task. This system uses an array of three fixed 'wing' cameras and two cameras mounted as a stereo pair on a robot arm, which permits them to be aimed at specific objects of interest from good view angles. Processing is performed by custom image-processing hardware and a general purpose computer for high-level functions. The image-processing hardware, originally IMFEX (for Image Feature Extractor) and being upgraded to PIFEX (for Programmable Image Feature Extractor) is capable of large numbers of operations on images and on image-like arrays of data. Acquisition utilizes image locations and velocities of features extracted by the feature extractor to determine the 3-dimensional position, orientation, velocity and angular velocity of an object.

PIFEX has been described in more detail elsewhere [1][2].

The organization of the acquisition and tracking system is shown in Figure 1. The Feature Tracker detects features in the images from each camera, tracks them as they move over time, smooths their two-dimensional positions, and differentiates the positions to obtain their two-dimensional velocities in the image plane. When enough features are being tracked, the Motion Stereo module uses the information from all of the cameras for some particular time to compute the partial three-dimensional information. The Stereo Matcher refines this information and computes estimates of the scale factor and bias. It uses a general matching process based on a probabilistic search. In this process, features from one camera are matched one at a time to features from another camera in order to build a search tree. For each combination of trial matches, a least-squares adjustment is done for the scale factor and bias that produces the best agreement of the matched features. The Model Matcher matches the three-dimensional feature positions (and any other feature information available) to those of the object model in order to determine the three-dimensional position and orientation of the object [3]. Meanwhile, the Feature Tracker, running concurrently with the other modules, still has been tracking the features (those that have remained visible). The latest positions of these features, together with the information from the model matcher that indicates which object features they match, are used by the Tracking Initializer to update the object position and orientation to the time of this most recent data. The position, orientation, velocity, angular velocity, and their covariance matrix from the Tracking Initializer are used as initial conditions in the Object Tracker. It rapidly and

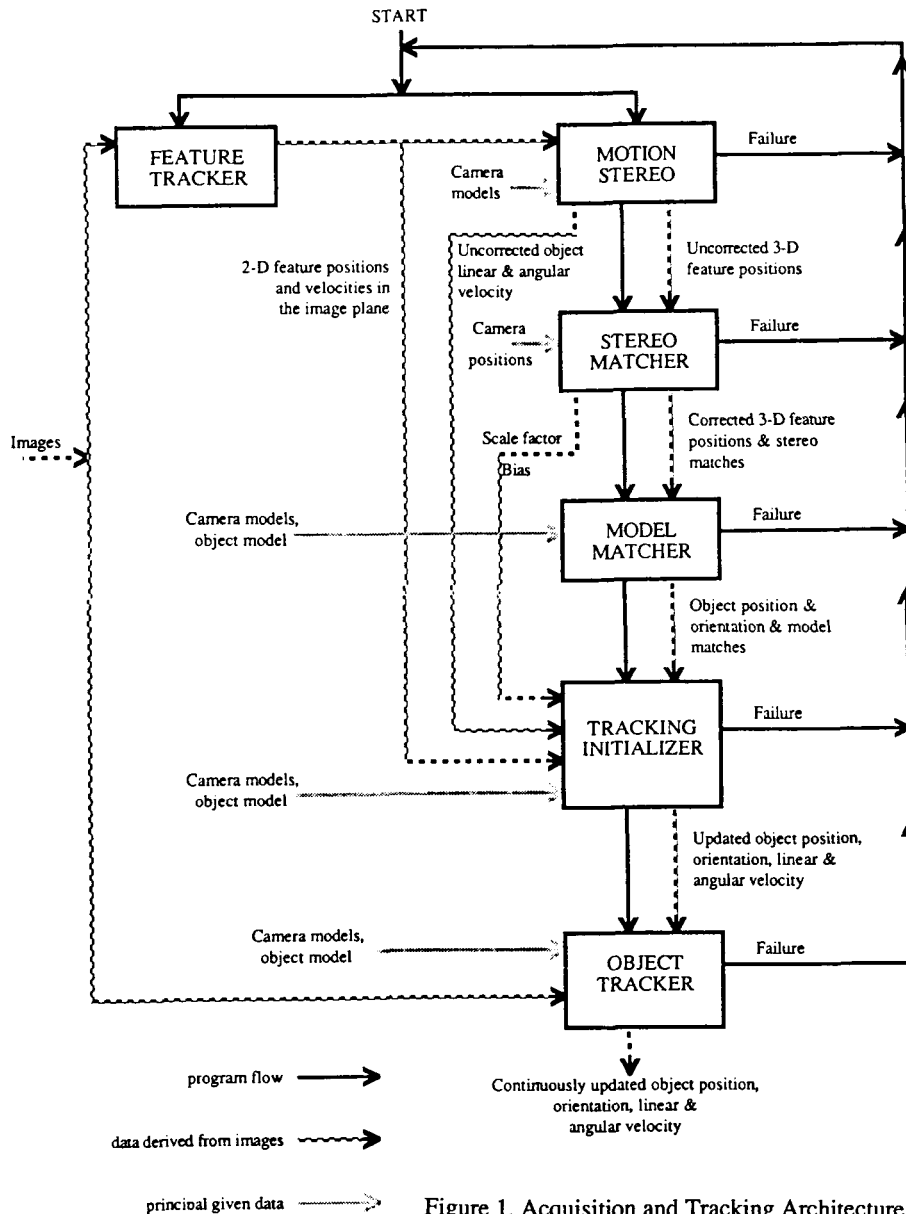


Figure 1. Acquisition and Tracking Architecture

accurately updates this information. Currently, the features that it looks for in the images are the object edges. Using edges produces more complete information than using vertices. Edges can be used easily here, because the one-dimensional information from edge elements suffices once the approximate object position and orientation are known [4].

More complete descriptions of the acquisition and tracking system have been published elsewhere [5].

VISUAL NAVIGATION AND HAZARD AVOIDANCE

Because of the long signal time to Mars (anywhere from 6 minutes to 45 minutes for a round trip at the speed of light), it is impractical to have a rover on Mars (the nearest candidate for a planetary rover) that is teleoperated from Earth (that is, one in which every individual movement would be controlled by a human being). Therefore, some autonomy on the rover is

needed. On the other hand, a highly autonomous rover (which could travel safely over long distances for many days in unfamiliar territory without guidance from Earth and obtain samples on its own) is beyond the present state of the art of artificial intelligence, and thus can be ruled out for a rover launched before the year 2000.

Semiautonomous navigation is an intermediate between these two extremes. In this technique, local paths are planned autonomously using images obtained on the vehicle, but they are guided by global routes planned less frequently by human beings using a topographic map, which is obtained from images captured from a satellite orbiting Mars. The orbiter could be a precursor mission which would map a large area of Mars in advance, or it could be part of the same mission and map areas only as they are needed. As commanded from Earth, the orbiter would take a stereo pair of pictures (by taking the two pictures at different points in the orbit) of an area to be traversed (if this area is not already mapped). These

pictures might have a resolution of about one meter, although poorer resolution could be used. The pictures are sent to Earth, where they are used by a human operator (perhaps with computer assistance) to designate an approximate path for the vehicle to follow, designed to avoid large obstacles, dangerous areas, and dead-end paths. This path and a topographic map for the surrounding area are sent from Earth to the rover. This process repeats as needed, perhaps once for each traverse between sites where experiments are to be done, or perhaps once per day or so on long traverses.

The sequence of operations taking place on Mars is as follows. The rover views the local scene and, by means discussed below, computes a local topographic map. This map is matched to the local portion of the global map sent from Earth, as constrained by knowledge of the rover's current position from other navigation devices or previous positions, in order to determine the accurate rover position and to register the local map to the global map. The local map (from the rover's sensors) and the global map (from the Earth) are then combined to form a revised map that has high resolution in the vicinity of the rover. This map is analyzed by computation on the rover to determine the safe areas over which to drive. A new path then is computed, revising the approximate path sent from Earth, since with the local high resolution map small obstacles can be seen which might have been missed in the low-resolution pictures used on Earth. Using the revised path, the rover then drives ahead a short distance (perhaps ten meters), and the process repeats.

With the computing power that it will be practical to put on a Mars rover in the 1990's, the computations needed to process a stereo pair of images and perform the other calculations needed may require roughly 60 seconds. If these are needed every 10 meters and it takes the rover 30 seconds to drive 10 meters, the resulting average rate of travel is 10 meters every 90 seconds, which is 11 cm/sec or 10 km/day. If a ten-kilometer path is designated from Earth each time, only one communication per day is needed, and the rover could continue to drive all night, using strobe lights for illumination. On the other hand, the method is more reliable than autonomous operation, because of the human guidance and the overview that the orbital data provides.

There are several types of computations that need to be done on the rover (or on a Mars orbiter in constant communication with the rover). These include the computation of a depth map, the computation of a topographic map, the matching of this map to the global data base and merging with it, analyzing the traversability of the area, planning a path, and monitoring the execution of the path. Some of these now will be discussed in more detail.

The first step in the processing on the rover is the production of the depth map (the distances to densely packed points over the field of view of the sensing device). One way of obtaining this is with a scanning laser range finder, which produces the depth map directly. Another way is to use two or more cameras for stereo vision. By the usual stereo process of matching and triangulation the depth map is computed. Other computer vision techniques, such as shape from shading and texture analysis, can aid in this process. Each approach has advantages and disadvantages. Stereo vision usually is more accurate at close ranges but less accurate at long ranges than laser range finders. On the other hand, laser range finders are limited in the maximum range at which they are effective. Laser range finders tend to make fewer errors than stereo vision, but each can fail to produce results under different conditions. Stereo vision and other computer vision techniques are attractive since stereo cameras will be available

for scientific sample designation or core locating purposes in any event. Also, solid state cameras are small, have no moving parts, and take less power than laser scanners. Stereo matching is a computation-intensive process which is benefiting greatly from recent advances in microelectronic fabrication, while scanning mirrors remain prone to mechanical problems. Most likely, a rover should use both a scanning laser range finder and stereo cameras, to produce the best results by combining their measurements and to provide reliability in case of failure.

Recent research efforts at JPL have studied various types of stereo matching. First, we have explored the use of sophisticated statistical algorithms to reduce the error rates of two-camera stereo. Second, we are exploring the use of additional cameras and multiple cross-correlation to reduce the error rate (and possibly reduce the computational load, due to the need to match smaller patches of the images in order to achieve a given level of reliability). By using a linear array of multiple cameras which are mounted parallel so that matching between cameras is along corresponding scan lines, special hardware could be built to implement matching algorithms at frame rate. Lastly, techniques of multiresolution pyramid decomposition have been employed (where a succession of low-pass and band-pass images are produced from the original image). In this multiresolution technique, objects are matched at low resolution, and then these matches are used to guide the search at the higher resolutions. All of these techniques show promise in producing reliable depth maps and they can be combined in various ways. All of these techniques use small image patches, which are correlated between images. These are called area-based techniques, and differ from the feature-based techniques (using edges or vertices) used in vision for space telerobots. The primary difference results from the fact that spacecraft (and man-made objects in general) have a relatively small number of well-defined visual features, while natural terrain has a very large number of edges and vertices, and so is not compactly represented by simple feature extraction.

Once acquired, the depth map is transformed into an elevation map (altitudes for densely but unequally spaced horizontal positions). An important issue is whether to keep the data in the iconic form of the elevation map, in which case the topographic map sent from Earth also would be in this form, or to reduce the data to a more symbolic form.

In the iconic case, the elevation map from one view is merged with the elevation map in the data base by a process of correlation and averaging, which also produces the best estimate of vehicle position as that which produces the best correlation. (Information other than elevations, such as reflectance, could be used also.) However, this computation is more complicated than ordinary correlation because the points are not equally spaced, there may be significant uncertainties in their horizontal positions, and there may be occasional mistakes in the stereo data.

In the symbolic case, some description of objects in the scene would be extracted, for example ellipsoidal approximations of rocks [6] together with descriptions of ground slope. The same type of description would be developed from the orbital images on Earth and sent to the rover, and the matching and merging process would use these symbolic descriptions [7]. Here the techniques of vision for natural terrains begin to converge with the techniques for space telerobotics, in that the symbolic representations can be viewed as a form of feature extraction. However, these features are much more complex (e.g. 'rock' or 'crater') than those used for man-made objects ('edges' or 'vertices'). Thus low-level processing and special

hardware are not generally applicable to this type of feature extraction.

With either kind of description, the local data are merged with the global data base to produce an updated data base. In some cases, each new view would be merged immediately with the global data base. However, in some cases, the matching process may not be able to correlate accurately with the global data base because of the lack of prominent features, but there may be enough smaller features to correlate with the high-resolution views seen previously from nearby locations. Therefore, a local data base could be built up by merging several local views. Then when sufficiently prominent features are encountered to match well to the global data base, the local data base would be merged with it. In general, there could be a hierarchy of data bases produced in this manner.

Traversability can be determined by analyzing the data base to determine the slope and roughness of the ground at each horizontal position. This can be done by local least-square fits of planar or other surfaces and analysis of the residuals. A way of doing this for the iconic representation will be tried in the current JPL project. (If the data base is in symbolic form, the information may already be there in the form needed.)

More complete descriptions of the Mars Rover local navigation and hazard avoidance process have been published elsewhere [8].

CONCLUSIONS

Machine vision will be an important element of both space telerobots and planetary rovers. Generally, the vision systems of space telerobots will use feature extractors to generate a reduced representation of the scene, and feature-based matching in multiple cameras to generate 3-D representations. Planetary rovers, on the other hand, will use area-based scene matchers (as well as active techniques such as laser scanning, which are less useful on orbital tasks due to the highly absorptive and specular surfaces employed) to determine the 3-D geometry of the scene. Once the 3-D geometry is known, space telerobots will generally try to recognize known objects or generic classes of items, such as fasteners. Planetary rovers may never need to assign symbolic names to objects in a scene-- a map of elevation, slope, roughness, estimated surface friction and load-bearing strength may be all that is needed. It is only a more sophisticated rover which needs to reason about landslides, unstable rocks, or 'box canyons' that may need any symbolic representation at all. Thus it may be that the two types of vision system remain quite distinct in their development, hardware, and implementation for many years to come.

ACKNOWLEDGMENT

The machine vision systems described in this paper have been created by the staff of the Robotics Systems Research, Implementation, and Integration group at JPL. The algorithms described are almost entirely the creation of staff member Dr. Donald B. Gennery. The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

REFERENCES

1. D. B. Gennery and B. Wilcox, "PIFEX: An Advanced Programmable Pipelined-Image Processor," Jet Propulsion Laboratory Publication 84-97, 1984.

2. D. B. Gennery and B. Wilcox, "A Pipelined Processor for Low-Level Vision," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* San Francisco, CA, June 1985, pp. 608-613.
3. D. B. Gennery, "A Feature-Based Scene Matcher," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, August 1981, pp. 667-673.
4. D. B. Gennery, "Tracking Known Three-Dimensional Objects," *Proceedings of the AAAI Second National Conference on Artificial Intelligence*, Pittsburgh PA, August 1982, pp. 13-17.
5. D. B. Gennery, "Stereo Vision for the Acquisition and Tracking of Moving Three-Dimensional Object," in *Techniques for 3-D Machine Perception* (A. Rosenfeld, ed.), North Holland, 1985.
6. D. B. Gennery, 'Object Detection and Measurement Using Stereo Vision,' *Proc. Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 320-327 (1979).
7. D. B. Gennery, 'A Feature-Based Scene Matcher,' *Proc. Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, pp. 667-673 (1981).
8. B. Wilcox and D. Gennery, 'A Mars Rover for the 1990's', *Journal of the British Interplanetary Society*, Vol. 40, No. 10, pp. 483-488, Oct 1987.

**ORIGINAL PAGE IS
OF POOR QUALITY**

VOICE CONTROL OF COMPLEX WORKSTATIONS

Jeffrey L. Scruggs

Texas Instruments, Inc.
Speech And Image Understanding Lab
Computer Science Center
P.O. Box 655474, MS 238
Dallas, Texas 75265

ABSTRACT

This paper describes the use of a speaker-dependent connected word recognition system to control an Air Traffic Control (ATC) demonstration workstation, and the work which went into developing that speech system. The workstation with speech recognition was demonstrated live at an Air Traffic Controller's Association convention in 1987. The paper discusses the purpose of the demonstration workstation, and highlights the development of the speech interface. This includes: a brief description of the speech hardware and software, and overview of the speech driven workstation functions, a description of the speech vocabulary/grammar, and details the enrollment and training procedures used in preparing the controllers for the demonstrations. Although no quantitative results are available, the paper discusses the potential benefits of using voice as an interface to this type of workstation, and highlights limitations of the current speech technology and where more work is required.

INTRODUCTION

For many years, speech has been recognized as one of the preferred man-machine interfaces. Within the last decade, with the advent of low-cost speech processing hardware and software, we have begun to see commercial applications which utilize speech as an interface between man and machines. There have been many successful systems providing voice response applications; for example, voice mail. Applications using speech recognition have generally been limited to areas where the vocabulary required to interact with the system has been small, and where the words can be spoken in isolation. Many of these successes have come in the areas of factory quality inspections and inventory control.

In the last 3 or 4 years, the speech industry has begun to address the problems associated with using speech to interface with more complex tasks. Some of these tasks have included: voice control of AFTI F-16 cockpit systems, dictation using a voice actuated typewriter (VAT), voice control of stock market order entry stations, and medical transcription terminals. All of these systems require higher levels of speech recognition performance than the earlier applications. These new applications require larger vocabularies, more connected speech capabilities, and easier training mechanisms. In addition, these new applications continue to require system recognition accuracies of greater than 95%. The efforts to meet the needs of these more complex tasks have met with varying degrees of technical success, but none of these systems have yet achieved widespread commercial success to date.

This paper describes an effort to use a commercially available speech recognition product from Texas Instruments as an interface to a complex workstation, an Air Traffic Controllers Workstation. This was a joint effort between the Ground Systems Group of Hughes Aircraft and the Computer Science Center Speech And Image Understanding Lab of Texas Instruments, Inc. The goal was to develop a speech interface for Hughes-designed demonstration ATC workstation. The workstation was displayed at the 1987 Air Traffic Controllers Association Convention in Los Angeles, and at 1987 Radio Technical Commission for Aeronautics in Washington DC. The speech system was developed for demonstration purposes only.

OVERVIEW OF THE SYSTEM

Figure 1 shows the main hardware components of the demonstration workstation. The controller's console was composed of a 20-inch square color display on which the operator could view the air traffic in his own and adjoining sectors. The console incorporated several interface technologies including a keyboard and track ball (used in current workstations), as well as a touch panel and speech recognition interface. Any of these devices could be used interchangeably by the controller to manage the console. A Sun workstation was used as the console controller during the demonstration, while a PC containing the TI speech recognition system was the other major piece of hardware. The PC was connected to the Sun via a RS-422 serial communication link. In order to provide realistic data during the demonstrations, the Sun workstation had been preloaded with a scenario from the Los Angeles International Airport area. The microphone cable was patched directly from the console to the speech hardware.

The purpose of the demonstration was to show how the use of various interface technologies made it easier for the controller to manage his or her workstation. There was no simulation of the link between the aircraft and the ground. Instead, the controller just managed the workstation in front of him. During the demos, the airspace scenario was free-running on the console, while the controller demonstrated the features of the man-machine interface while using the various input devices.

DESCRIPTION OF THE SPEECH SYSTEM

The speech system used was the Texas Instruments LR2000 recognition system. The hardware is a single board option for IBM PC's and compatibles based on the TMS32010 digital signal processing chip. The board is a flexible speech peripheral capable of performing a wide variety of speech processing tasks including: record / playback, text-to-speech, speech recognition (both isolated and connected speech), and speaker verification. In addition, an application software development kit is available to allow users to write custom applications utilizing any of these speech capabilities.

Figure 2 shows a block diagram of the speech recognition process. The lowest level is a speaker-dependent word hypothesizer which has two inputs: the real-time speech input, and previously stored vocabulary word templates. As the user speaks, his input speech is compared against the templates; when a match is found the hypothesizer outputs a result to the second level of the system. This second level is the sentence recognizer. This subsystem compares the output of the word hypothesizer with a previously defined grammar structure, and

outputs recognized sentences to an application program on the PC. The grammar structure is a finite-state grammar which describes all the valid sentences in the application domain. The vocabulary and grammar used in the ATC demonstration are described in the next section.

The advantages of this two-level decision structure are two-fold: first, the robustness of the recognition is improved since more global knowledge of the application environment is available at the recognizer level (in the form of the application grammar), and two, by using improved training procedures the users can speak to the system using connected speech.

DESCRIPTION OF THE VOCABULARY AND GRAMMAR

The task of determining where to use speech as an interface was a cooperative effort including Hughes human factors experts, the controllers who would be demonstrating the systems, and the Texas Instruments speech application developer. There were three principal areas where speech was considered: voice recognition driven by the radio uplink to aircraft, voice recognition to control the workstation console, and speech synthesis to notify controllers of conditions requiring attention. Since the scenario to be demonstrated did not include a simulation of the radio uplink, that area was rejected. The other two areas were both considered very promising for using speech, and were both within the capabilities of TI's speech product. However, due to time constraints in preparing for the convention, only the console control voice recognition was actually implemented.

The voice commands fell into two major categories: console display control and aircraft situation acknowledgement. The console display control functions were concerned with how the data was displayed on the main console color display. These included displaying data from other control sectors, changing the display range in miles, and highlighting critical flight data elements for specific aircraft. The aircraft situation acknowledgement commands included: acknowledging aircraft alerts, acknowledging flight plan postings, marking handoff of aircraft between sectors, and assigning altitudes, beacon codes, and preferential routes. A vocabulary of 94 words was defined to provide these functions. Table 1 provides a list of the vocabulary used in the demonstration.

As previously mentioned, the LR2000 recognition system requires both a vocabulary list and a application grammar. Definition of a grammar for the ATC workstation application did not prove very difficult since the controllers are already trained to use a standard "language" when controlling their airspace. Figure 3 shows a

portion of the system grammar describing the acknowledgement of alerts and the highlighting of flight data elements (FDE). The symbol <flid> indicates that the controller could at that point in the grammar say any of the flight identifiers which were available in the scenario and had been programmed into the grammar. The symbol <1 - 8> indicates that the controller could say any digit from one to eight. Due to the limited nature of the demonstration, the possible flight identifiers were restricted to those occurring during the scenario. This restriction was also required due to the vocabulary size limitations of the TI speech system. This size limitation is related to the processing power and memory space available on the speech hardware, and is not a physical limitation of the recognition algorithm.

ENROLLMENT AND TRAINING STRATEGY

Because the speech recognizer was speaker dependent, the system had to be trained to recognize each individual speaker. For the three day convention, 8 controllers were chosen to demonstrate the ATC workstation. Each controller was required to enroll the complete 94 word vocabulary.

The enrollment strategy for the LR2000 system is a two-step process where each word is said once as part of a sentence and once in isolation. These two "templates" are then used to create a single recognition template for that word. The resulting template incorporates "coarticulation" effects which normally prevent speech recognition systems from being used with connected or conversational speech. The initial template creation took approximately 45 minutes for the 94 word vocabulary. Subsequently, each controller maintained their templates by periodically repeating a set of sentences which included all 94 words of the vocabulary. This update process required about ten minutes per repetition.

In order to ensure good performance at the convention, the updates were performed at different times of the day over several weeks. In this way, each template included the daily variations which all of us have, along with any long-term variations which might appear. During this process, several of the controllers had colds or allergies; the effects of which were included in the update process. The updates were supervised by one of the controllers who was trained to monitor the template creation and update function. By the time of the convention, the 8 controllers had updated each word at least 7 times. In addition, at the convention, each of the controllers was again updated to accommodate the new acoustic environment in the convention hall.

RESULTS AND CONCLUSIONS

The goal of the speech demonstration was met. The system demonstrated that today's speech recognition systems, in particular TI's LR2000 connected word recognition, are capable of being used in a complex workstation environment. The system was demonstrated with speech by the 8 controllers for three days approximately 6 hours each day. During that time, the controllers were able to demonstrate the operation of the ATC console using all the various interface technologies, including voice. The overall impression of both the viewing audience and the controllers using the system was that a speech recognition system providing connected speech recognition can be a useful part of an improved man-machine interface for advanced ATC workstation.

While this demonstration was a success, there are many areas that must be considered by a system designer before a speech interface is actually implemented. Most of today's speech systems are speaker-dependent, and thus require user enrollment and training. Many of the systems available do not have connected speech capabilities, they require a pause to be inserted between each word. TI's LR2000 connected word recognition system is an exception. Most systems also have limitations on the number of words which can be recognized at one time. In addition to these concerns, the system designer also must look at the recognition accuracy which is required and how recognition error recovery will be handled.

This is not to say that speech systems do not have a role to play. Recognition provides an excellent interface for tasks where an operator's hands and eyes are busy. In addition, in systems where the operator is required to manage a large variety and amount of data, the addition of speech as an alternative input device may provide an improved man-machine interface.

Training systems utilizing recognition could provide high quality, lower cost operator training where the recognizer would be used to determine the correctness of communication between an operator and other people. An example might be to use recognition to mimic the role of the pilot in aircraft under the control of an air traffic controller. Other speech technologies can also be used to provide more effective workstations. Speech output can provide audible warning or help messages. Speaker verification can be used to ensure that only authorized personnel log into a workstation.

In conclusion, today's speech systems can provide

for a more effective man-machine interface in the complex workstations required to manage complex tasks.

ACKNOWLEDGEMENTS

The author would like to acknowledge the efforts of the many individuals at Hughes Aircraft Co. who contributed their time and energies to making the speech portion of the demonstration a success. In particular, he would like to recognize Augie Beining, for his part in having the speech interface included in the first place. In addition, the author thanks the controllers who worked to learn the system and did such a fine job in demonstrating it. In particular, he would like to recognize Larry Suppan who was responsible for the enrollment and training of the rest of the controllers.

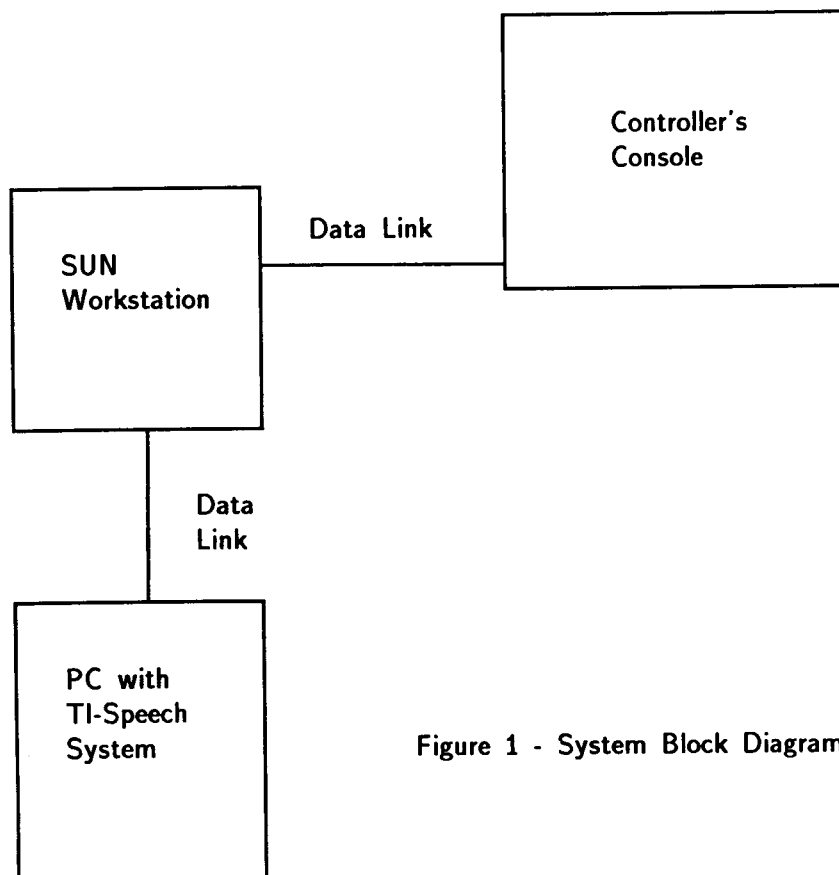


Figure 1 - System Block Diagram

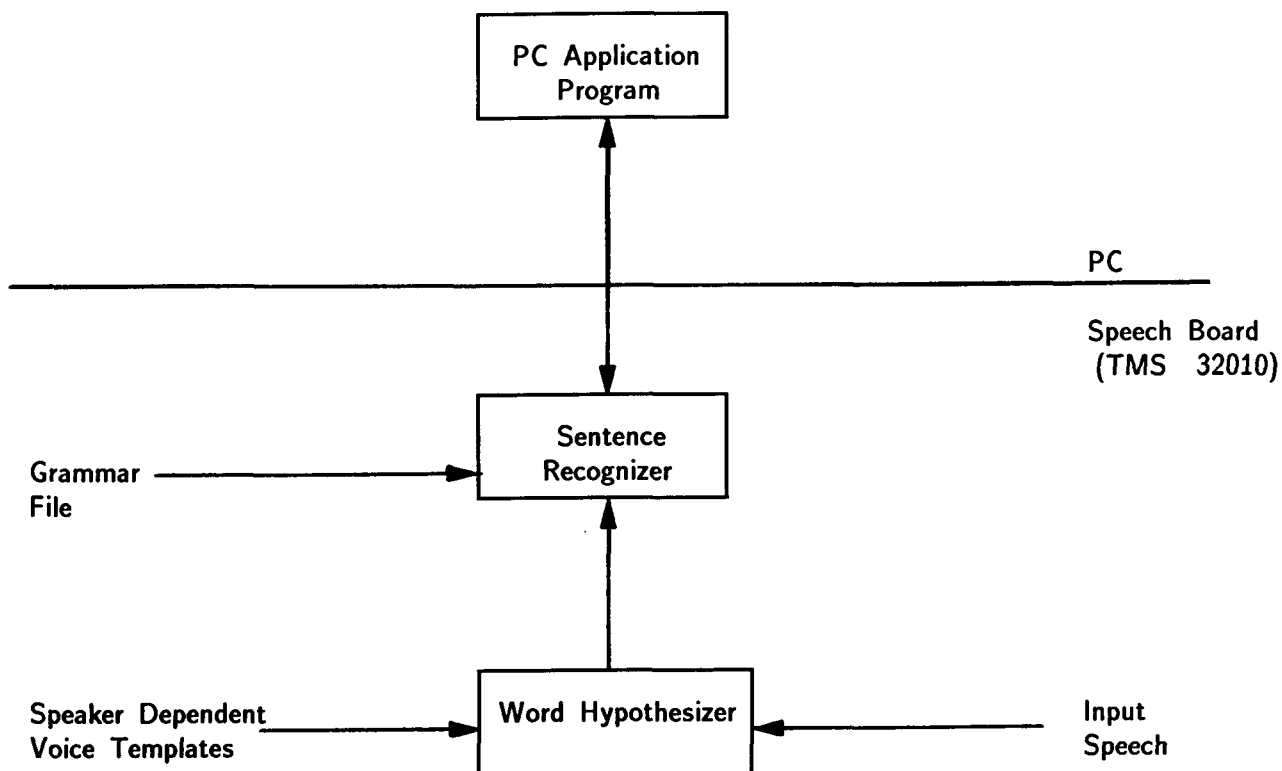
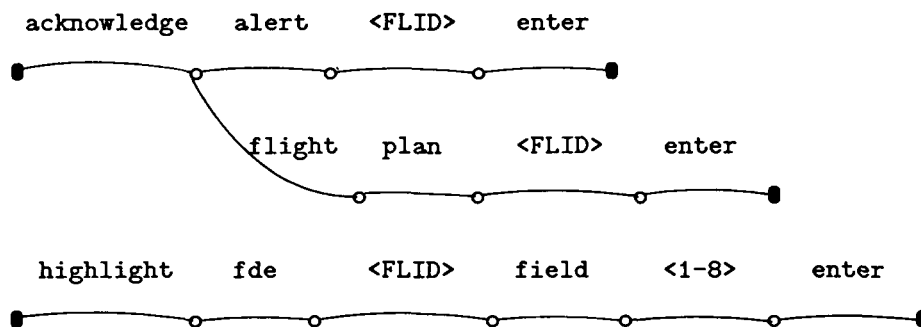


Figure 2 - LR 2000 Block Diagram

ZERO	OH	ONE	TWO
THREE	FOUR	FIVE	SIX
SEVEN	EIGHT	NINE	NINER
TEN	"EELEVEN"	"UHLEVEN"	THIRTEEN
FIFTEEN	SEVENTEEN	EIGHTEEN	TWENTY
THIRTY	FORTY	FIFTY	ENTER
"ENNER"	THOUSAND	AIR FORCE	AMERICAN
DELTA	UNITED	PSA	NOVEMBER
ROMEO	SNOW	NAVY	MIKE
WHISKEY	DROP	TRACK	FLIGHT
PLAN	HISTORIES	SITUATION	INSET
RANGE	MARKS	START	VELOCITY
VECTOR	MILE	HIGHLIGHT	FDE
ACKNOWLEDGE	MOVE	ALERT	QUICK
LOOK	DATA	BLOCK	HANDOFF
ACCEPT	FIELD	TYPE	INITIATE
SECTOR	CANCEL	POINT	OUT
ASSIGN	ALTITUDE	REPORTED	INTERIM
BEACON	CODE	PREFERENTIAL	ROUTE
CHANGE	FREQUENCY	LEVEL	CESSNA
SPEED	FIX	VENTURA	TIME
ESTIMATED	DIRECT	SANTA	BARBARA
PALMDALE	CORRECTION	NORTH	SOUTH
EAST	WEST		

TABLE 1 - VOCABULARY LIST



(continued for remainder of grammar)

Example Sentences: "Acknowledge Alert American Ten Enter"

"Highlight FDE November 9871 Delta Field 3 Enter"

FIGURE 3 - PORTION OF APPLICATION GRAMMAR

A MULTI-SENSOR SYSTEM FOR ROBOTICS PROXIMITY OPERATIONS

J.B. Cheatham, C.K. Wu, P.L. Weiland
Rice University
Mechanical Engineering & Materials Science
Houston, Texas 77251-1892

T.F. Cleghorn
NASA/Johnson Space Center - FM 7
Houston, Texas 77058

ABSTRACT

Robots without sensors can perform only simple repetitive tasks and cannot cope with unplanned events. A multi-sensor system is needed for a robot to locate a target, move into its neighborhood and perform operations in contact with the object. This paper describes systems that can be used for such tasks.

INTRODUCTION

An integrated robotic system capable of performing tasks in which the target is in an arbitrary location and orientation is described in this paper. This system consists of a PUMA 560 robot manipulator, a vision system with a camera mounted over the robotic work area, a force/moment sensor, a laser range finder, and an infrared proximity sensor. Two laboratory experiments were performed successfully using this system. One experiment simulates a simple industrial assembly task and the other permits battery replacement in a model of the Solarmax satellite. In the first experiment, the vision system is used to locate randomly placed parts. In the second, a laser range finder is combined with the infrared proximity sensor to locate the satellite model. These two 'visual' systems provide redundant information about an unknown environment. This redundancy can be used to check or confirm the visual information provided by either system. Fine motion control is achieved by employing the force/moment transducer and the infrared proximity sensor.

Sensors are monitored and controlled by an expert system written in CLIPS (C Language Integrated Production System). CLIPS is a rule-based language written in C. All rules are sequenced by a priority rating. The priority rating is set by the rule ordering method [1] which arranges all rules in one long priority list. The rule with the highest priority, appearing earliest in the list, is triggered first.

SYSTEM OVERVIEW

The RTX robot, manufactured by UMI, is a six degree-of-freedom robot with five revolute axes and one translational axis. This manipulator is operated under program control using the C language. Belt drives are used for the shoulder, elbow and yaw joints and bevel gears for the roll and pitch axes. Accuracy of repetitive movements is within 0.5 mm.

Figure 1 depicts the system configuration. The PUMA 560 is an industrial grade, six degree-of-freedom robot. This manipulator can be operated interactively or under program control using the VAL II language. A NAMCO Lasernet scanner attached to the first joint of the PUMA provides a vertical search arc of 90 degrees to a distance of twenty feet. Rotating this device about the vertical first joint of the PUMA effectively provides a complete search of the robot's workspace.

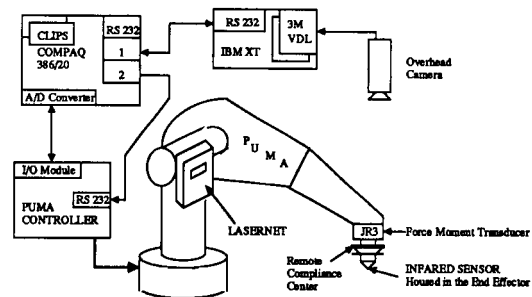


Figure 1. System configuration

A specially designed tape that reflects incoming light back at the incident angle is attached to the target. The returning pulse is detected by the Lasernetet, which signals that the target has been located. An infrared proximity sensor, built into the end effector, determines the target orientation by examining a known surface of the target.

A JR3 force/moment transducer, mounted on the end effector of the PUMA, provides a means for controlling the motion of the robot while maneuvering the objects. The JR3, an industrial grade force/moment measuring device [2], consists of a six degree-of-freedom force sensor and an electronics enclosure that contains a 12-bit ana-

log to digital converter. Resolution of the force measurements is 1 part out of 4096 of the full scale, (± 25 lbs along the x and y axis, ± 50 lbs along the z-axis, and ± 75 in-lbs for moments). The digital data output from the A/D board is transferred to the PUMA controller via the direct memory access (DMA) bus. A VAL II subroutine, provided by the JR3 company, can be called in a program for measuring the forces and moments exerted on the end effector.

A 3M vision development language (VDL) system with a camera mounted over the robotic work area provides a high-speed vision development workstation for an IBM XT computer. This system includes a $512 \times 512 \times 8$ digitizer with two 256 Kbyte frame buffers and a signal processing board for image acquisition and processing. A wide variety of vision algorithms can be implemented interactively using macros, programmed with a command interpreter called VDL-BASIC, or programmed using C language supported by a C command subroutine library. Work described in this paper uses the C programming capabilities.

INTELLIGENT VISION-ASSISTED ASSEMBLY TASK

The first experiment simulates an assembly task involving two robots. A number of objects of various sizes and shapes are randomly placed in the work area (see Figure 2). A camera over the work area captures images of the objects and data relating to the images are passed from the vision system to the expert system for identification. One by one the objects are retrieved by the RTX robot and placed in a position for use by a PUMA 560 robot in an assembly task. The objects are picked up by the Puma and inserted into holes of the proper sizes and shapes under force-moment control.

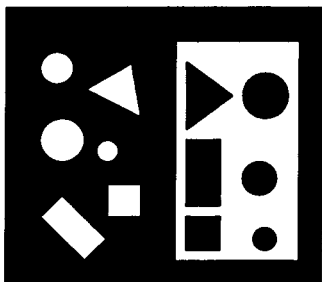


Figure 2. Top view of the objects and receptacle

An overhead camera connected to the 3M VDL vision system provides digitized images of the unknown targets. The vision program consists of a calibration subroutine, an image acquisition subroutine, a target extraction subroutine, and a properties measurement subroutine. Video images of the objects are digitized and thresholded into binary images by the image acquisition routine. Targets are represented as white objects on a black background in the binary image. The target extraction routine searches the binary image for a single white

object and encloses the object within a window. The contour of the windowed object is then traced and the locations of the corners are determined [3]. This procedure is repeated for all objects in the viewing plane.

Processed vision data are transmitted from the vision system to the expert system on a Compaq 386 through an asynchronous communication line. The CLIPS program analyzes the vision data and determines the shapes, sizes and locations of the objects. This program consists of rules to identify the objects and to sort the targets by their geometric properties. It then transmits the position data to the RTX.

The RTX is programmed in C to perform the disk retrieval task. Center of gravity and orientation data of an object are developed by the expert system program. These data provide the necessary information for the RTX to pick up the object and move it to a predefined location. Upon completion of this task the control program is notified of the task completion and the RTX returns to its home position. The PUMA 560, programmed in VAL II, is instructed by the expert system program to retrieve the disk and move it to the taught position for the assembly work station. Insertion of the object is accomplished using force/moment feedback provided by the JR3 transducer. After successfully inserting an object a signal is transmitted from the PUMA to the expert system program. This process is repeated for the remaining objects. Upon completion of the insertion of the last object, both robots return to home positions.

BATTERY REPLACEMENT IN A SATELLITE MODEL

The goal of the second experiment is to develop a procedure for changing batteries in a satellite. A PUMA robot is programmed to perform this task autonomously. The robot carries a tray with tools and fresh batteries in a known position with respect to the robot. The location and orientation of the satellite containing the battery compartment is unknown.

An essential part of this experiment is to locate the satellite model without using the vision system. A scanning laser range finder is used to locate a reflective material attached to the target and to determine the coordinates of the object. A proximity sensor, which is housed in the end effector (see Figure 3), is then used to determine the orientation of the target. This system provides an alternative means of acquiring positioning information and is relatively inexpensive compared with a vision system [4].

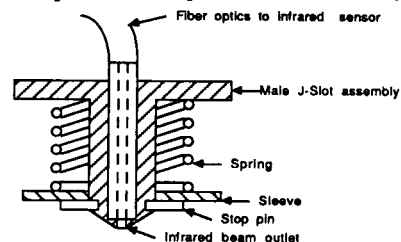


Figure 3. Infrared sensor housed in an Interchangeable end effector

The Lasernet senses only its reflected light. Commercially available reflective tape can reflect incident light back to the origin at angles up to 70 degree from the surface normal. However, variations in this angle are observed. The width of the target must be known for the Lasernet data to be used in determining distance. An object in a horizontal plane can readily be located by using a flat, circular reflective tape. As the object's orientation becomes more complex, so does the target selection. Ideally, a spherical target will provide the most uniform projection at any orientation. In this experiment, a spherical reflective target was constructed out of the reflective tape adhered to a golf ball.

During operation, the PUMA rotates the Lasernet about its first joint searching for the spherical target attached to the satellite. When the target is located the Lasernet sends a signal to the PUMA which records its first joint angle. The location of the spherical target is then computed from the range and angle data from the Lasernet and the PUMA's first joint angle [4].

Once the spherical reflector has been located, the IR proximity sensor is used to conduct a search for a smaller circular reflector (see Figure 4). The proximity sensor receives a reflected "on" signal within 2 cm of the aluminum plate on the satellite model and up to 50 cm from the reflective tape targets. The end effector is moved in a circular search pattern of radius equal to the known distance from the sphere. By locating the two reflective targets a line is defined on which the satellite model is located. The proximity sensor is then moved midway between the reflectors and lowered to detect the object's surface. A search for the edges of the aluminum plate defines its orientation. The sensor receives a positive impulse while over the surface, but loses the signal as it crosses over the edge. The orientation of the satellite model is found by tracing one of the edges. A false edge is used to determine which edge is found. A non-reflective strip is placed down the length of the object. When first seen by the sensor it appears as an edge. The sensor will continue onward after the 'edge' is found and regain contact with the surface. This on-off-on signal will signify the location of the false edge. Knowing these points the orientation of the object can be established.

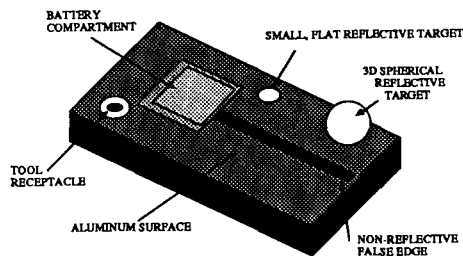


Figure 4. Object/Target Configuration

Once the location of the satellite model is completely determined, the battery replacement task may proceed. The PUMA engages an electric cover actuator

to turn a power screw which opens and closes the door over the battery compartment. A J-Slot type assembly is used to pick up both the battery packs and the electric cover actuator [5]. Figure 5 depicts the electric actuator with the J-Slot. The retrieval and insertion of batteries into the battery compartment are performed under position and force control using the JR3 sensor.

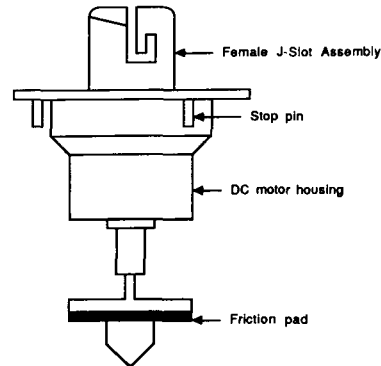


Figure 5. Electric actuator with J-Slot assembly

CONCLUSIONS

Two systems have been described that can be used for robot proximity operations. A vision system combined with a force sensor has been used to locate and identify objects then retrieve and insert them into receptacles of appropriate size and shape. An expert system written in CLIPS interprets the vision data and sends instructions to two robots used in the assembly task.

The second system used a laser range finder, an infrared proximity sensor, a force/moment transducer and special reflective targets to locate a satellite model and replace a battery pack in the model. A combination of both system can provide redundancy that could permit improved reliability for space robotics activities.

ACKNOWLEDGMENTS

The authors appreciate support of this work by NASA/JSC under Grant NAG-208 and RICIS contract NCC9-16.

References

- [1] Giarratano, J.C., CLIPS User's Guide, A. I. Section, NASA/JSC, Houston, September, 1986.
- [2] JR3, Universal Force/Moment Sensor System, Operational Manual, JR3, Inc., Woodland, CA, 1985.
- [3] Cheatham, J.B., Wu, C.K., Chen, Y.C. and Cleghorn, T.F., "Interpretation of Robot Vision Images via An Expert System", ASME International Computers in Engineering Conference, San Francisco, CA, 1988.

[4] Cheatham, J.B., Weiland, P.L. and Wu, C.K., "A Sensor System for Determining Position and Orientation of Robot Targets", 3rd International Conference on CAD/CAM Robotics and Factories of the Future, Southfield, MICH, 1988.

[5] Armstrong, A., Dunn, P., Hekma-Wierda, D., Hodge, A. and Howard, D., Design, Construction and Implementation of a Robotic System for Replacing Batteries, Senior Design Project Report, MEMS Dept., Rice Univ., May, 1987.

ORIGINAL PAGE IS
OF POOR QUALITY

A METHODOLOGY FOR AUTOMATION AND ROBOTICS EVALUATION
APPLIED TO THE SPACE STATION TELEROBOTIC SERVICER

Jeffrey H. Smith, Max Gyamfi, Kent Volkmer, Wayne Zimmerman

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109

ABSTRACT

The efforts of a recent study aimed at identifying key issues and trade-offs associated with using a Flight Telerobotic Services (FTS) to aid in Space Station assembly-phase tasks is described. The use of automation and robotic (A&R) technologies for large space systems would involve a substitution of automation capabilities for human EVA or IVA activities. A methodology is presented that incorporates assessment of candidate assembly-phase tasks, telerobotic performance capabilities, development costs, and effects of operational constraints (STS, attached payload, and proximity operations). Changes in the region of cost-effectiveness are examined under a variety of system design assumptions.

A discussion of issues is presented with focus on three roles the FTS might serve: (1) as a research-oriented testbed to learn more about space usage of telerobotics; (2) as a research based testbed having an experimental demonstration orientation with limited assembly and servicing applications; or (3) as an operational system to augment EVA and to aid the construction of the Space Station and to reduce the programmatic (schedule) risk by increasing the flexibility of mission operations.

INTRODUCTION

There has been continuing interest in the use of telerobotics for Space Station activities as a possible means for reducing EVA/IVA activities and operations costs, increasing safety, and improving the technology base and spin-off potential of telerobotics (NASA/JSC, January 15, 1987; National Academy of Sciences, 1986). A large-scale analysis of the Space Station assembly phase by the Critical Evaluation Task Force (CETF, 1986) in the Fall of 1986 resulted in the concern that the required EVA hours for assembly exceeded on-orbit EVA time constraints. This concern resulted in the recommendation that a Flight Telerobot Servicer (FTS) be used as an option for possible use starting at First Element Launch (FEL--the first flight in the Space Station construction phase). While the CETF recognized that an FTS could make a substantial contribution to reducing EVA during the construction phase, it was not clear whether such a system

built with an inherent technical risk would be cost-effective. This question motivated the need for the methodology presented herein.

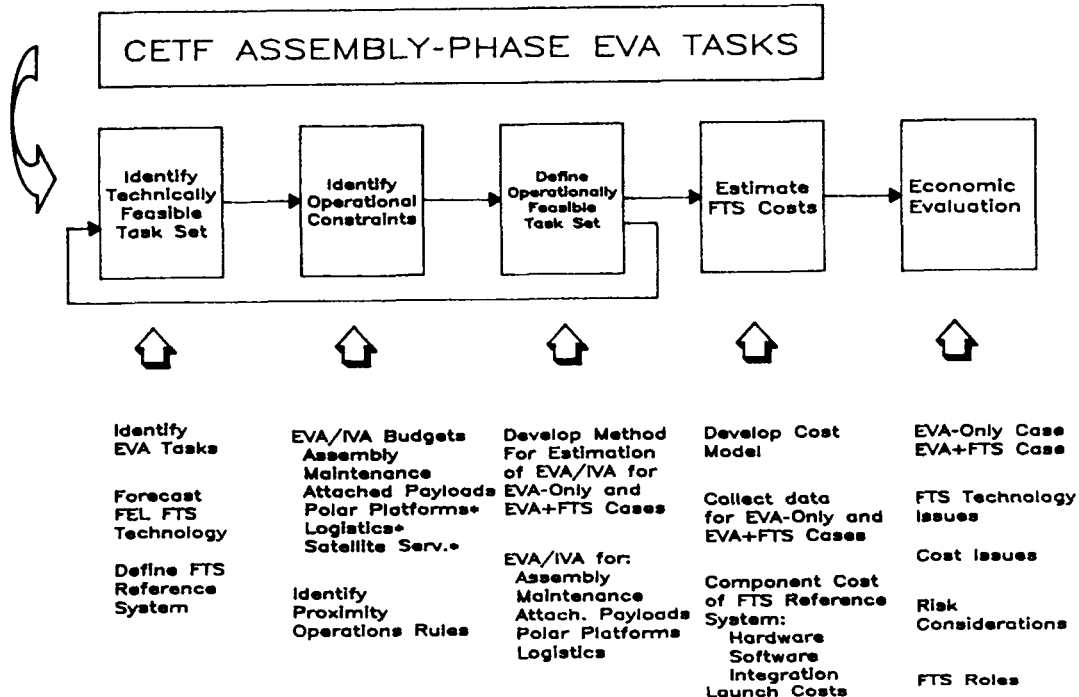
A key milestone for Space Station assembly, the Permanently Manned Configuration (PMC), is the point at which astronauts can reside for long periods on orbit without returning to earth with the Space Shuttle. The period from FEL to PMC is severely constrained for EVA resources, due to the short (Shuttle-based) time intervals for construction (approximately one week). There is a need to displace EVA resources; where "need" is defined as an FTS capability to reduce crew-EVA time so that absolute Shuttle-based EVA limits are not exceeded. Furthermore, the FTS must accomplish this reduction in a manner that is at least as cost-effective and reliable as available alternatives. The degree of mismatch between task activities and EVA requirements during the construction phase results in excessive EVA (which is expensive and hazardous), additional power requirements for the Space Station (to support the additional crew to perform the EVA tasks), and potential additional STS flights to "make up" shortages of EVA time. After PMC, the value of the FTS can be argued to depend on a more complex set of considerations: life-cycle cost, productivity gains, safety improvements, technology spin-offs, and other factors. This paper focuses on cost factors: considerations such as safety and technology spin-off benefits were not explicitly addressed.

The purpose of this paper is to present an approach for assessing the feasibility of utilizing telerobotic systems in the space environment and present the results of an application of the methodology to the Space Station. The results and design issues encountered are based on a recent investigation by the authors (Smith, et al., 1987).

APPROACH FOR COMPARING SPACE STATION TELEROBOTICS OPTIONS

A comparison of Space Station telerobotics options involves many complex factors. The objective is to provide a systems-level methodology that addresses the important components affecting the value of an FTS to the assembly phase. The approach, illustrated in Figure 1, is delineated below.

ORIGINAL PAGE IS
OF POOR QUALITY



*Examined but not included in the final results

Figure 1. FTS Assembly Phase Study Approach

Technically Feasible Task Set

A technically feasible task set is derived from a list of task activities (based on CETF, contractor studies, etc.) in the areas of assembly, payload servicing, and maintenance. In parallel, an FTS "Reference System" is defined based on a review of potential technologies required to implement the tasks, and, that will be available by FEL (i.e., 1996). For the Space Station application, an FTS Reference System is derived that could perform a subset of the assembly phase tasks at a level of technical readiness corresponding to the FEL date (although the technically feasible task set and corresponding Reference System may initially be somewhat incompatible with total system constraints). However, the purpose of this step is to capture the possible extent of task requirements and capabilities before applying operational constraints to insure the final reference configuration functionality is synchronized with all system constraints.

Operational Constraints

The operational constraints consist of EVA and IVA budgets and proximity operations rules that reduce the technically feasible task set to an operationally feasible task set. The following categories of activities were examined to estimate the EVA and IVA times for two cases: EVA-Only (no FTS) and EVA+FTS (FTS present) (NASA/JSC: March 1986; November 1986; January 8, 1987).

- (1) Assembly tasks
- (2) Maintenance tasks
- (3) Attached payload setup and servicing tasks

The operational constraints are overlaid on the technically feasible tasks set to derive an operationally feasible task set, and the FTS Reference System definition is revised to reflect the operational constraints. The EVA and IVA times for the two cases were estimated by flight, category (assembly, maintenance, and attached payloads), and year during the construction phase to measure the savings accrued by the FTS during the operations phase (Machell, 1986, McDonnell-Douglas, 1986).

Flight Telerobotic Servicer (FTS) Reference System

To assess the benefits and costs of an FTS, a design concept is required to focus the required technology capabilities and estimate costs. An FTS system is needed that is appropriate for specific EVA tasks required for assembly and operation of the Space Station between FEL and IOC. Such an FTS forecast addresses the availability of critical constituent technologies required at FEL, and highlights essential support characteristics such as FTS reliability, maintenance, and associated logistics support. Selection of technology capabilities must also consider schedule requirements (when must the system be operational), technology and system integration, system verification and testing, and system integration into Space Station operations. The study objective was to identify a low-risk, technically feasible FTS

Reference System that could be ready by FEL and could perform a set of operationally feasible tasks during the Space Station construction phase.

Before developing a reference configuration, the functional requirements for the system as a whole must be understood (NASA/JPL, 1986). As the desired functional capabilities are explored, obvious conflicts between FEL functions and technologies are identified and used as discriminators to maintain the list of functional requirements within the realm of feasibility (e.g., tasks requiring a considerable amount of on-line planning for fault management, or a large degree of dexterous manipulation, would not have the commensurate technology in place to meet the task needs). Tasks considered technically feasible in the FEL to IOC time frame include (1) basic assembly tasks such as pallet handling, worksite preparation, or truss construction in a well-defined, almost industrial robotic type environment, (2) simple orbital replaceable unit (ORU) change-out and inspection type tasks on payloads, (3) Space Station support tasks such as surface cleaning and inspection, (4) pick-and-place type logistic tasks such as transferring components or fluid consumables from the Shuttle to the Station, and (5) other support such as transporting equipment from one place to another, holding equipment in place while it is worked on by EVA astronauts, or providing on-site visual monitoring of an EVA task.

Given a set of possible technically feasible tasks, telerobot technologies are matched against those tasks. The key variables in selecting the technologies are:

- (1) Level of technology readiness (i.e., with FEL being the deadline for delivery)
- (2) Degree of system integration
- (3) Accuracy and repeatability requirements
- (4) Reliability
- (5) Retrofit considerations for future capabilities growth

An important element of technology readiness is whether the technology has the potential for being flight-qualified by FEL (Zimmerman and Marzwell, 1985). Empirical data gathered on system development elapsed time from concept to full operational capability (i.e., space qualification) suggest a time frame between five and ten years for moderately complex systems, and ten to twenty years for complex systems. Therefore, considering the FTS system as a moderate-to-complex design with an appropriate logistics support program in place by FEL, it was determined that likely FTS robotic technologies would probably not exceed the present state-of-the-art unless an aggressively funded flight test program or other experience gathering mechanism were introduced to reduce risk.

The next step in identifying a reference system is to develop an array of "strawman" FTS configurations that contain the required robotic technologies while meeting the projected task requirements. It was understood that the same tasks could be done in different ways, depending on the FTS configuration. For example, employing a more

sophisticated configuration such as a mobile FTS versus a fixed FTS offers greater flexibility and a wider range of applicability in component handling types of tasks. By developing several strawman configurations, it is possible to understand how other factors such as operational constraints (e.g., FTS operations in proximity to EVA) might influence the selection of particular configuration over another. It is likely that EVA-FTS proximity operations constraints could severely limit the possibility of any type of free-flying FTS being deployed. System control constraints imposed by the task environment and available technology could also limit the ability of the system to compensate for self-induced or environmentally induced dynamic disturbances or changes in the pre-planned task environment. For control and vision purposes, the approach is to select the most reasonable reference configuration from the subset of strawman designs. This study, supporting an FEL in the early 1990's, resulted in a reference design having a fixed base in which the fixed base is fastened and the FTS is transported manually to the base using the Shuttle RMS or the MSC where it is connected for operations.

Assembly Phase EVA and IVA Resource Estimates

Due to large uncertainties in some of the data components, ranges are used to bound the results (a formal analysis of these uncertainties was not performed). The total EVA times per flight-interval for the EVA-Only and EVA+FTS cases are illustrated in Figure 2 using low-range EVA estimates for construction, maintenance, and attached payloads. The low-range values represent the lowest estimates for the EVA range obtained by adding all the low values together. A similar procedure was used for the high-range estimates. The aim was to bound the actual values by examining the extreme low and high values. The estimates of Figure 2 are troubling. The estimated EVA required on five flights prior to PMC exceeds the budgeted amounts of 24 hours. This finding supports the argument that the CETF assembly sequence does not manifest within the CETF constraints for at least three early flights. This is due primarily to construction on Flights 1 and 2 and maintenance and attached payload contributions on subsequent flights. The implication is that for the CETF design to work, one or more shuttle flights must be added, the current shuttle flights must be extended (unlikely), or there must be a re-manifesting of construction EVA to meet the constraints. It is the cost of additional shuttle flights that dominates the cost-effectiveness of the FTS.

RESULTS: ASSEMBLY PHASE COMPARISON WITH AND WITHOUT THE FTS

An economic model was developed to examine the cost-effectiveness of the FTS Reference System and to determine whether the FTS could be cost-effective during the construction phase. The Net Savings model is:

Space Station Assembly-Phase EVA EVA-Only versus EVA+FTS Case Low-EVA Estimates

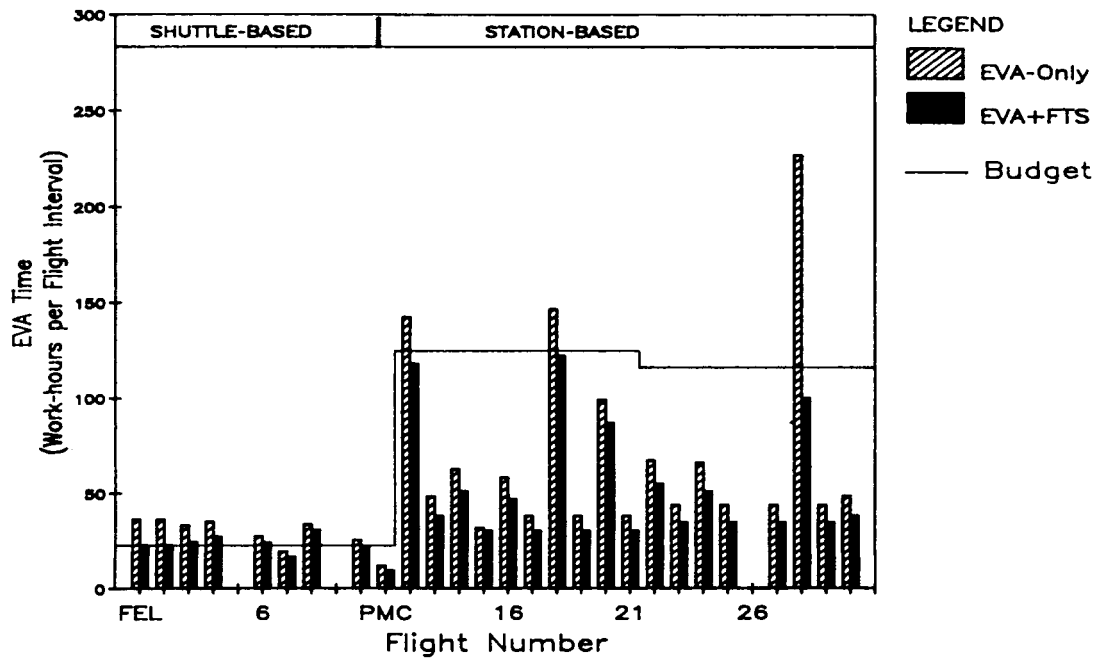


Figure 2. Construction Phase EVA Estimates--Low-Range EVA Values

Net Savings Due to the FTS Reference System =
(Operations and Maintenance Cost of EVA-Only
Case minus
Operations and Maintenance Cost of EVA+FTS
Case) minus
Investment Cost of the FTS.

If the Net Savings is positive, the FTS Reference System is cost-effective. The use of this approach required a cost estimate of the FTS Reference System and a bottom-up cost (component-by-component) estimate was made using the component list for the FTS Reference System (Smith, et.al., 1987). An estimate of \$277 million (M) to \$304 M was obtained for the FTS (excluding non-prime costs--the costs of managing the prime contracts and spares costs). The costs and benefits for the development of the FTS up to the completion of the construction phase were examined. At issue was the feasibility of using the FTS to assist in the assembly process. Thus, benefits to users or the Station after the construction phase were not examined. FTS ground operations costs were included using estimates of FTS operating costs. Using these cost estimates and the EVA and IVA profiles, a series of analyses were performed to determine the feasible region for the FTS Reference System.

The results indicate that during the assembly phase the major tradeoff evolves around the cost of the FTS and the cost-per-flight of the STS.

Because of cases where the estimated EVA exceeds the budget of 24 hours during FEL to PMC, additional flights must be added to make up the difference. The cost of any added flights as a major factor in the cost-effectiveness of the FTS. Figure 3 presents one such trade-off region using the low-range estimates of EVA/IVA and the FTS cost over a range of STS costs per flight from \$105M to \$178M. It is difficult to determine an estimate for STS prices. Estimates have ranged from below \$100M to \$150M during the pre-Challenger era. The assumption was made that the price will be higher in the post-Challenger era due to increased safety and reliability requirements, component re-designs, and quality control constraints. However, a range of price curves is presented to provide a generalized result. The FTS cost ranges from a low \$232M (NASA estimate) to \$340M (National Research Council, 1987); the end points were selected merely to bound the trade-off region. The area in the center of the region bounds the FTS Reference System estimated costs. As an example, if we assume a STS cost of \$150M, the FTS will break even if it can be built for a cost of \$292M or less. If the FTS costs more than \$292M, it will not be cost-effective (unless the STS price is actually higher). For the other points on any of these curves, the estimated net savings can be read from the axis on the left.

Also, note the term "Mixed Manifesting" on Figure 3. This refers to assumptions made regarding how

FTS VS. STS TRADE-OFF REGION Low EVA Estimates/Mixed Manifesting

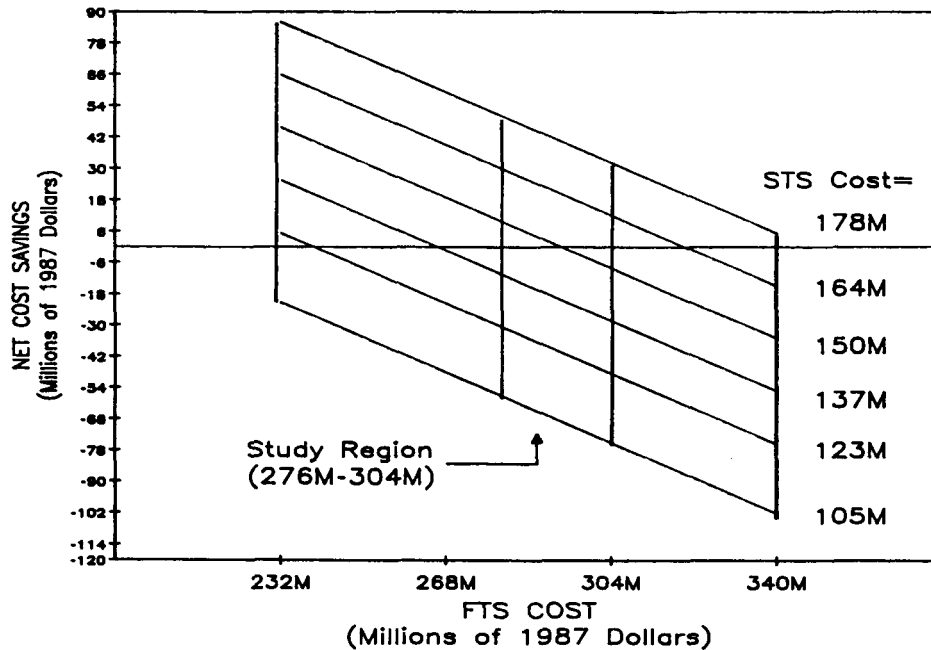


Figure 3. FTS versus STS Trade-Off Region

excess EVA is remanifested on subsequent flights if an additional flight is required. Note that as manifesting becomes inflexible, the FTS cost effectiveness region moves up (toward cost-effective) and as manifesting becomes flexible, the FTS cost-effectiveness moves down (toward less cost-effective).

If the scenario is moved toward the flexible manifesting assumption, the trade-off region moves down (toward less cost-effective) because fewer overall flights are required. If the scenario is moved toward the inflexible manifesting assumption, the region moves up (more STS flights are required). Furthermore, as the differences between the number of additional flights in the EVA-Only case and the EVA+FTS cases (if any) becomes larger, the width or spacing between the curves also becomes larger. The constant slope of the curves (approximately -0.75) is an indication that for each reduction in FTS cost of one dollar, there is an increase in net savings of only \$0.75. The remaining 25% is the delivery cost and the effects of discounting.

The region in Figure 3 is for the low-range EVA values. If the high-range EVA values are used, the region moves down. Similarly, as the estimated cost of the FTS increases, cost-effectiveness drops (the region shifts downward).

Another parameter of interest is the EVA cost per hour used to estimate the cost of EVA hours used. As with the STS cost, the estimation of such a value is difficult. To examine the sensitivity of the results to EVA cost per hour, three cases were examined using \$45,000 (\$45K), \$35K, and \$25K per

hour. Note the apparent insensitivity of the region to this parameter. This is due to the magnitudes of the numbers between the FTS and STS costs. A decrease in the cost per hour simply places less value on the resource benefits the FTS can displace and thus makes the FTS region move down. At least for the assembly phase, it appears that EVA cost (summed over a reasonably short period of time) are dominated by the Shuttle ferrying costs. This result does not imply that life cycle Space Station EVA costs will be equally insignificant. The discount rate used in the above results is the Office of Management and Budget (OMB) value of 10% used for cost-benefit analysis on government projects. The effect of varying the discount rate was also examined using a 6% rate. The effect of reducing the discount rate is to move the trade-off region up significantly. This indicates that a lower discount rate would have a significant impact on improving the cost-effectiveness of the FTS.

DESIGN ISSUES AND IMPLICATIONS

A key design issue is to identify the key attributes of the FTS program that affect the trade-offs to be made between the numerous users the robotic system faces. If the attribute to be maximized is the commercial benefit to be derived from technology advances (i.e., spin-off potential), then a different value equation (than net savings) will need to be constructed in order to accommodate these technologies to be stimulated, and thus the activities that the FTS can be used to demonstrate. It was assumed here that the objective was to maximize the overall value of the FTS to the Station. Thus, technology develop-

ORIGINAL PAGE IS OF POOR QUALITY

ment programs need to be instituted that enable FTS performance upgrades in areas that directly enhance FTS value to the Station. This could be done by identifying high-payoff applications amenable to acceptable-risk FTS system configurations. This assumption need not minimize the role of the FTS program in stimulating automation and robotic (A&R) technology development since both terrestrial spin-off and Station benefits can accrue from development of intelligently selected advanced technologies.

The current study was performed over a period of time in which the Station design moved from the CETF concept to an abbreviated Phase I configuration. However, because the STS-based EVA activity is still highly constrained in the Phase I case, the results are likely to be robust.

It is important to note that whether or not the FTS is cost-effective for the assembly phase, there are still legitimate uses under a number of scenarios. If the FTS is not cost-effective, it could still serve as a research and development testbed for post-IOC applications. If it is cost-effective, it could be used as an applications-oriented tool. Earlier studies have highlighted some of these role differences varying from a low-cost orbiter-based operational system to a space-based testbed for evolving telerobotics technologies (Goddard, 1986). Although there is a range between an applications-oriented versus a demonstration-oriented FTS, even if marginally cost-effective, the FTS could still serve as a backup, that could reduce schedule risks by providing a flexible assembly/servicing option for some additional EVA activity, if needed. This is an important design issue because it must be shown that a net risk reduction exists. Situations where the added risk of a large robot system (that could fail into a dangerous mode, or require extensive maintenance or EVA attention) must be understood prior to dedication of the system to an operational role. A robotic system can play a testbed or demonstration role in order to gather experience with on-orbit operations at a point where the design of the operational system can be modified. The interfaces between the human operators, the equipment, and the task requirements can be refined or revised to make better use of the synergistic potential of re-designed tasks coupled with FTS capabilities specifically designed for those tasks. If it is assumed that FTS operations are terminated at IOC, or that the FTS is not used for Station operations but rather for research and demonstration purposes, then there are other benefits this paper made no attempt to qualify. One class of benefits is the development of "lessons learned" that can be utilized to develop a future FTS that does play an integral role in a wider variety of Station and on-orbit operations. Such experience would provide a valuable database for guiding the design of future tasks and FTS capabilities.

Note that the analysis performed herein is inherently conservative. Limiting the time frame of the analysis to the construction phase underestimates the actual benefits of an FTS by excluding any

post-IOC benefits. If the FTS is assumed to continue operations after IOC, the FTS feasibility region will tend to move upward (towards more feasible) for all cases. This paper presents a single solution out of many possible ones, and the results described are by no means optimal. The FTS option selected here was based on an analysis of estimated task requirements and estimated functional requirements. The focus was to identify the components that ought to be examined when comparing FTS options. Nonetheless, a number of recommendations are made.

There is a need to examine the effects of risk in these comparisons (Smith, et al., 1985). Cost risk can be viewed directly using the net savings, or operations and maintenance (O&M), equations to generate breakeven estimates for net savings and O&M costs. Then, as assumptions of the problem (such as software/integration costs) are varied, the impact on the breakeven point can be computed. Technical risk can also be studied in terms of the uncertainties in performance and reliability. In addition, the effects of specific risk elements, such as the effects of introducing suits requiring no pre-breathe step, EVA overhead, and the effects on EVA if such a suit is not ready on schedule, could be singled out. An understanding of the risk and uncertainty effects would show how the FTS could help reduce program risk by adding flexibility to operations planning and contingency planning--especially during FEL-PMC. There is value and benefit of having an FTS for the flexibility it provides for dealing with unscheduled events. Further study of the risk elements would quantify those benefits.

Additional study is also needed for the allocation of automation and robotic functions. Very different results can be achieved by locating such functions on the ground. With improved autonomous operations, Station IVA could be reduced. One question is whether to pursue advanced and potentially technically risky autonomous or semi-autonomous options versus an investment in on-the-ground remote telerobot operation capability. Such activity would identify the issues related to the human factors and control technology problems of dealing with time delays in teleoperation feedback. It may be possible to mitigate the problems of such time delays with predictive control and large scale dynamic task environment simulation technologies. The present paper has shown the magnitudes of the savings to be potentially large enough that a dedicated FTS relay system to provide near real-time response might be an alternative worth considering. This will depend on the potential for extending the displacement of IVA and EVA task times while minimizing the technical risk of developing the system. If extended operations can be performed from the ground, the risk of requiring additional flights may be reduced and provide a schedule margin during the early FEL-PMC period when assembly elements must be completed within fixed, short term flight periods or risk mission failure. The area of allocation of autonomous and robotic functions and resources needs further examination to help designers select whether A&R upgrades are performed on the Station, incorporated

ORIGINAL PAGE IS OF POOR QUALITY

into the FTS, or operated on the ground (see Zimmerman, et al., 1985).

A related allocation problem that requires further understanding is the allocation of work among and between multiple robots (FTS, RMS, MSC, etc.) and crew EVA (co-EVA). Data on performance time ratios for such mixed tasks should be collected for a variety of tasks using neutral buoyancy studies and (eventually) on-orbit experience. The proximity operations rules for such operations will also have to be identified in detail.

DISCUSSION AND CONCLUSIONS

A number of conclusions can be drawn, based on a CETF-derived (30-flight) construction phase. Noting that the results are conservative in that benefits were not considered; safety benefits were not considered; and the effects of the satellite servicing facility were not examined; the following conclusions were drawn:

- (1) The FTS Reference System identified herein appears to be technically feasible for development by FEL.
- (2) The FTS Reference System is cost-effective under a variety of conservative scenarios.
- (3) The STS cost is the primary factor for FTS cost effectiveness due to avoidance of extra STS flights, driven by EVA reductions.
- (4) The FTS is cost-effective at a 10% OMB discount rate but even more cost-effective at a 6% rate.
- (5) The assembly-phase is a maintenance problem (50% of total EVA is for maintenance versus 33% for construction). FEL-PMC is the primary construction problem.
- (6) The FTS Reference System defined here is most suitable for performing:
 - (a) Truss construction tasks
 - (b) Limited ORU replacement tasks
 - (c) Deployment of special equipment
 - (d) Pallet handling, loading, and unloading tasks

The potential exists for transferring some on-orbit tasks to ground operations given that appropriate technology and human engineering constraints are considered.

- (7) The total estimated cost of the FTS Reference System is \$277 to \$304M (does not include non-prime costs or spares).
- (8) There is a need for improved and more detailed data on task descriptions, timelines, manifests, etc. updated quarterly or semi-annually and available via electronic mail, for example.
- (9) A methodology for comparing automated and robotic options has been developed with specific applications to the FTS and its technical and cost feasibility for use during construction phase construction. Other A&R elements could be analyzed in a similar manner (see Zimmerman, Bard, Feinberg, 1985).

The approach described in this paper is intended to assist in the characterization of an assembly role for which an early robot or FTS might best be designed. Potential for cost-effective early operation argues for an FTS and host environment designed to facilitate performance of the selected FTS tasks. On the other hand, marginal early operating benefits suggest the option of treating the FTS initially as a testbed for development of advanced technologies that will later serve the Station in a more cost-effective manner.

A related issue is that of reliability, or more accurately, program confidence in the reliability of the FTS to perform tasks determined analytically to be cost-effective. The Advanced Technology Advisory Committee and Space Station work package contractors have been remarkably consistent in their conclusions regarding which tasks were within the capabilities of telerobotic devices. Program personnel, citing the criticality of early (pre-PMC) EVA tasks, are considerably more skeptical. The CETF, for example, ultimately based its results on the use of deployable utilities in preference to use of an FTS, on the grounds that on-orbit construction by telerobotic devices had never been attempted. This suggests that the subject of both ground and flight demonstrations of the FTS should be directed specifically toward whatever tasks the FTS might be applied to initially, particularly in cases of high task criticality.

Finally, multiple competing goals have been articulated for the mandated FTS development program and it is not clear that the program adequately addresses this issue. For example, the goal of increased Station productivity and decreased operational cost implies a high-reliability, low-technical risk, low-maintenance FTS that can be brought on-line early in the Station operating life. This approach cannot be easily reconciled with the aggressive station/FTS program schedule and less aggressive investment in A&R technology development. At the same time, it is clear from studies such as the CETF, that the "push-in here-pop-up there" EVA manifesting problem will not go away in the immediate future.

ACKNOWLEDGEMENTS

This paper presents research carried out by the Jet Propulsion Laboratory, California Institute of Technology, Space Station Project, which is an agreement under JPL Contract Number NAS 7-918.

REFERENCES

1. CETF, Critical Evaluation Task Force, NASA Langley Research Center, Langley, Virginia, August 20-September 14, 1986.
2. Machell, R.M., "Space Station EVA Time Requirements," McDonnell Douglas Astronautics Co., presented at the Critical Evaluation Task Force (CETF) Meeting, NASA Langley Research Center, Langley, Virginia, August 20-September 14, 1986.

**ORIGINAL PAGE IS
OF POOR QUALITY**

3. NAS, Report of the Committee on the Space Station of the National Research Council, National Academy of Sciences, National Academy Press, Washington, D.C., September 10, 1987.
4. NASA/JPL, Functional Requirements for the 1988 Telerobotics Testbed, JPL Document No. D-3693, Jet Propulsion Laboratory, Pasadena, California, October, 1986.
5. NASA/JSC, Space Station Mission Requirements Data Base, Johnson Space Center, Houston, Texas, March, 1986.
6. NASA/JSC, Space Station Program Definition and Requirements, Section 3: Space Station System Requirements, Revision 3, Document No. JSC 30000, Johnson Space Center, Houston, Texas, November 26, 1986.
7. NASA/JSC, Space Station Assembly and Maintenance Architectural Control Document, Document No. JSC 30502, Level B Change Request BJ020195, Johnson Space Center, Houston, Texas, January 8, 1987.
8. NASA/JSC, Flight Telerobotic Servicer Baseline Configuration Document, Document No. JSC 30255, Johnson Space Center, Houston, Texas, January 15, 1987.
9. Smith, J.H., Feinberg, A., Lee, T.S., Miles, R.F., Reiners, T., and Schwartz, D.L., Autonomy Evaluation Methodology: A Microcomputer Tool for Design Trade-Offs of Spacecraft Systems, Vols. I and II, JPL Document No. D-2761, Jet Propulsion Laboratory, Pasadena, California, November, 1985.
10. Smith, J.H., Gyamfi, M., Volkmer, K., and Zimmerman, W., The Space Station Assembly Phase: Flight Telerobotic Servicer Feasibility, JPL Document No. 1234, Jet Propulsion Laboratory, Pasadena, California, December, 1987.
11. Zimmerman, W.F. and Marzwell, N., "Space Station Level B Automation Technology Forecasting/Planning Structure," JPL White Paper prepared under sponsorship of the Johnson Space Center, Houston, Texas, April 30, 1985.
12. McDonnell-Douglas Astronautics Company, Assembly Sequence, presented at the Technical Integration Panel Meeting, Johnson Space Center, Houston, Texas, May 6-7, 1986.
13. Zimmerman, W.F., Bard, J., and Feinberg, A., Space Station Man-Machine Trade-Off Analysis, JPL Document No. 85-13, Jet Propulsion Laboratory, Pasadena, California, February 15, 1985.
14. Firschein, O., NASA Space Station Automation: AI-Based Technology Review, Prepared for NASA-Ames Research Center under Contract No. NAS2-11864, SRI International, Palo Alto, California, April 1, 1985.
15. McDonnell-Douglas Astronautics Company, The Human Role in Space (THURIS), Vol. II, Contract No. NAS8-35611, DPD No. 624, DR-4, October, 1984.
16. McDonnell-Douglas Astronautics Company, Automation and Robotics Plans (DR-17), Vol. II, Contract No. MDC H2036B, December, 1986.
17. NASA/JSC, Space Station Program/National Space Transportation System Interface Requirements, Document No. JSC 30503, Johnson Space Center, Houston, Texas, January 12, 1987.
18. Personal Communication--teleconference between J.H. Smith and C. Armstrong, Crew Systems, regarding proximity operations rules for an FTS using the STS Program Operational Flight Rules as a guide; Johnson Space Center, Houston, Texas, October 28, 1986.
19. Personal Communication--teleconference between J.H. Smith and D. Webb, regarding proximity operations rules, Johnson Space Center, Houston, Texas, October 28, 1986.
20. Personal Communication--teleconference between W.F. Zimmerman and C. Cole, Chief Automation Engineer, General Motors Corp., regarding task performance time ratios, Detroit, Michigan, January 30, 1987.
21. Weber, T., "Assembly-Phase Maintenance Analysis," Rockwell International, presented at the Space Station Configuration and Analysis Panel Meeting, Johnson Space Center, Houston, Texas, May 6-7, 1986.
22. Loyola, S., Mankins, J., and Wheeler, R., Space Station Mission Requirements Synthesis Study, Final Report, JPL Document No. JPL D-4267, Jet Propulsion Laboratory, Pasadena, California, October, 1986.
23. NASA/JPL, Space Station Polar Platform Payload Servicing Requirements, JPL Document No. JPL D-3177, Rev. A, Jet Propulsion Laboratory Pasadena, California, December, 1986.
24. COCOMO Expert System Software Costing Assistant, Level Five Research, Inc., 503 5th Ave., Indiatlantic, Florida, 32903.
25. Akkerman, J.W., Space Station Cost Management Process Requirements, JSC Document No. 30470, Johnson Space Center, Houston, Texas, December 30, 1986.
26. Grumman Space Systems, Space Station Assembly Study, Document No. V86-0555-001B, Grumman Corp., March, 1986.
27. Personal Communication--teleconference between M.A. Gyamfi and J. Oberight, regarding FTS operating cost, Goddard Space Flight Center, Greenbelt, Maryland, February 17, 1987.

28. NASA, Space Station Program Definition and Requirements, NASA (Draft) TBD, Washington, D.C., August 5, 1987.
29. Office of Management and Budget, Discount Rates to be Used in Evaluating Time-Distributed Costs and Benefits, OMB Circular A-94, March 27, 1972.
30. Grumman Aerospace Corp., Analysis of Remote Operating Systems for Space-Based Servicing Operations, Final Report, Document No. V84-1694-001B/NAS-17066, Report SA-ROS-RP-10, Grumman Corp., November 26, 1984.
31. Grumman Aerospace Corp., Telepresence Work System, System Definition Study, InterContract No., NAS 9-17229, Document No. V85-1383, Grumman Corp., October, 1985.
32. Cassingham, R., Space Station User Documentation: Lessons from the Space Shuttle, JPL Document No., JPL D-4487, Jet Propulsion Laboratory, Pasadena, California, June 30, 1987.
33. Meissinger, H.F., "Technology Requirements of Telerobotic Satellite Servicing," presented at the NASA/JPL Workshop on Space Telerobotics, Pasadena, California, January 20-22, 1987.
34. Smith, J.H., "A Microcomputer Tool for Design Trade-Offs," presented at the 53rd Military Operations Research Society Meeting, U.S. Air Force Academy, Colorado Springs, Colorado, June 25-27, 1985.
35. NASA/JSC, Flight Telerobotic Servicer Baseline Configuration Document, See Requirement No. 3.15.1.A, Document No. JSC 30255, Johnson Space Center, Houston, Texas, January 15, 1987.
36. Drysdale, A., Cost Factors Relating to Allocation of Crew Time, Document No. A91-F477-014 McDonnell-Douglas Astronautics Company, October, 1984.

SENSING HUMAN HAND MOTIONS FOR CONTROLLING DEXTEROUS ROBOTS

Beth A. Marcus Ph.D.

Philip J. Churchill

Arthur D. Little

Center for Product Development

20 Acorn Park

Cambridge MA, 02140

ABSTRACT:

The Dexterous Hand Master™ (DHM) system is designed to control dexterous robot hands such as the UTAH/MIT and Stanford/JPL hands. It is the first commercially available device which makes it possible to accurately and comfortably track the complex motion of the human finger joints. The DHM is adaptable to a wide variety of human hand sizes and shapes, throughout their full range of motion.

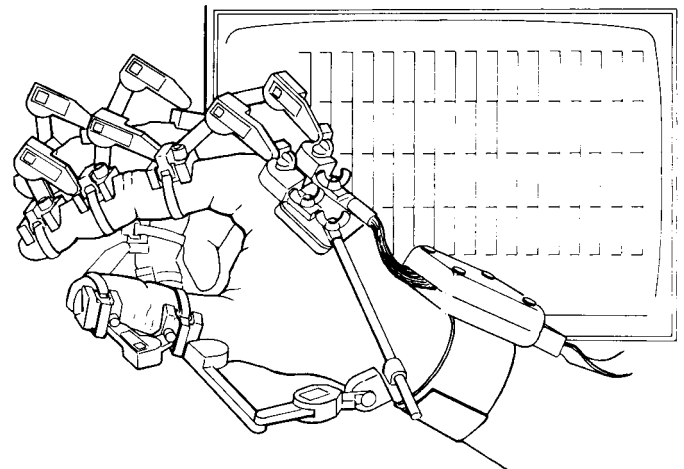


Figure 1- Dexterous Hand Master (DHM) system

BACKGROUND:

Laboratories around the country have begun to use dexterous robot hands such as the UTAH/MIT and the Stanford/JPL hands for robotic research applications. These hands are currently controlled through computer software, with little capability for teleoperation. As robotics technology has progressed, the dexterity of end-effectors has improved greatly, yet to make these dexterous hands widely and commercially applicable, an easy to use method for controlling them is required.

In January, we began a program to develop a sensing device capable of measuring human hand joint angles and translating them into position control commands. ADL selected an exoskeleton

design approach based on initial development work performed by Center for Engineering Design (CED) at the University of Utah in Salt Lake City, in conjunction with their dexterous robot hand program. Although their basic exoskeleton concept was viable for this application, several improvements were required to make this approach into a commercially available product.

With this starting point we developed a system which collects joint angle data on 16 finger joints, in real time, through the use of Hall Effect sensors and an AT compatible microcomputer. The result is a system which is accurate, light weight,

comfortable to use, and easily adjusted to fit a wide range of human hand sizes.(figure 1).

PROGRAM OBJECTIVE:

The goals of this program were to develop a system which could accurately transduce human joint motions and provide them as control signals to robots. In order to achieve this objective we set the following design goals to improve on the work of CED :

- Reduced Bulk,
- Reduced Weight,
- Improved Comfort,
- Improved Reliability,
- Improved Fit,
- Improved Attachment,
- Improved Resolution, and
- Addition of Calibration to Accommodate Hand Sizes.

SYSTEM DEVELOPMENT:

In order to achieve these goals we began with the selection of the sensor. The CED device, which was created primarily for testing the UTAH/MIT hand during development, used off the shelf potentiometers. Through experimentation with a variety of conventional, high precision, and conductive plastic potentiometers, we found that these sensors had the following disadvantages:

- Unacceptable levels of resistance to rotation (friction),
- Large package volume, and
- Signal output levels overly sensitive to noise, wear, and other environmental factors.

Therefore, we began considering other sensing methods and finally settled on Hall Effect sensors which had been successfully used for measurement of motion in other robotics applications (i.e. the UTAH/MIT hand, also developed at CED). A Hall Effect sensor is a small semiconductor device which changes its signal output voltage in proportion to the magnetic field it is experiencing. Its advantages include:

- non-contact measurements (no friction),
- small package,
- signal output not subject to many environmental influences, and
- available with amplification on the chip.

Using a magnet assembly designed by CED for the UTAH/MIT hand, we tested various Hall Effect sensors and selected a Honeywell product which featured built-in amplification and excellent temperature stability. We then experimentally determined the geometric relationship between the magnet and sensor which provided us with the desired response curve (figure 2). The important dimensions derived from these tests were the air-gap between the sensor and magnet faces, and the radial displacement of the magnet and sensor centerlines. In the proper configuration, the sensor output voltage will vary sinusoidally with the angular position of the magnet.

Once the sensor geometry was fixed, we moved on to the simultaneous development of the linkage mechanism and the electronics package. Through study of anthropomorphic data and experimental observations, we selected optimum linkage lengths for each joint of each finger. We built a single finger prototype using a planar linkage, with rigid attachment blocks, held to the fingers with Velcro™ straps. Extensive work with this prototype revealed three key problems:

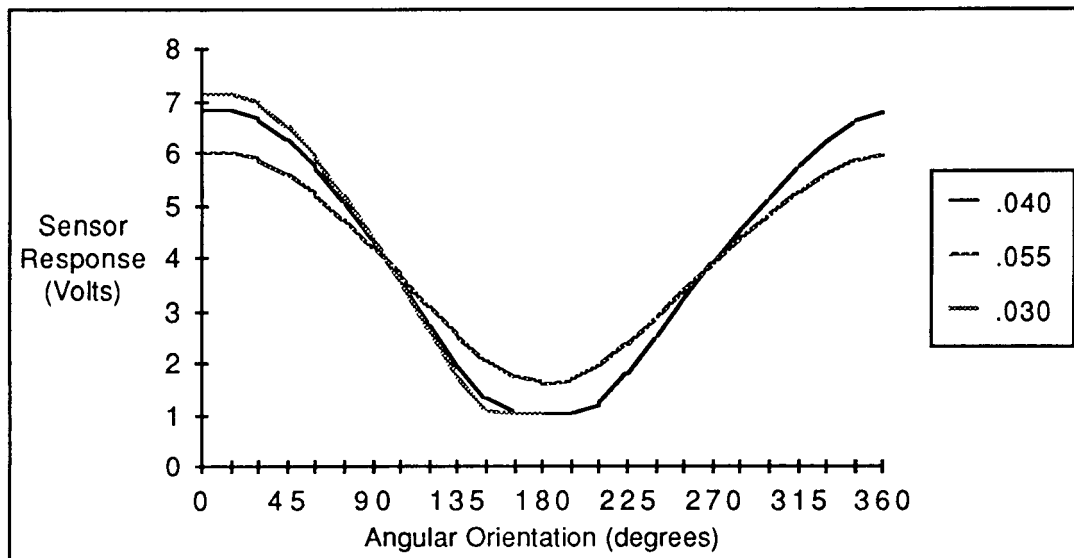


Figure 2- Hall Effect Sensor Response For Various Spacings

- Human fingers do not bend in a single plane (many people have noticeably crooked fingers),
- The relative orientation of the finger axes change continuously throughout the range of joint motion, and
- Finger segments grow and shrink substantially during bending,

The planar linkage therefore, was not acceptable because of its discomfort and tendency to shift at the attachment points. This led to pinching and measurement inaccuracy. Additionally, the solid attachment blocks would either flop about on the finger or restrict the range of motion of the finger depending on the angle of deflection (usually both).

We subsequently developed two features (figure 3) which eliminated these problems; the spring clip attachment, and the passive pivots. The spring clip allows a rigid strap (i.e. Velcro™) to change size as the finger bends, while keeping the attachment block firmly located on the finger segment. The passive pivot allows the finger to

bend freely, while maintaining the integrity of the sensor measurement.

Once the finger mechanisms were completed, we began solving the more difficult problem of measuring the thumb angles. The design of the distal joint linkage for the thumb was copied from the finger designs. The main problem we encountered was the measurement of the two proximal joints of the thumb. We concluded that it would not be possible to comfortably and repeatably secure an exoskeletal mechanism to the proximal link due to the large mass of soft tissue which makes up the base of the thumb and the unusual geometry of the joint itself. The solution we devised was to place two sensors on a linkage whose geometric relationship to the thumb joint motions was known. This solution requires a software algorithm which calculates both joint angles in real-time as a function of the two sensor inputs. In addition, we added another passive pivot to the thumb base to decouple several off - axis rotations which are not applicable to a robotic thumb command.

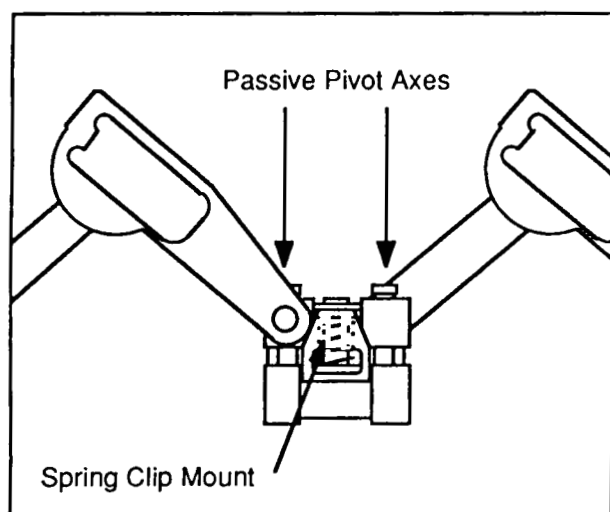


Figure 3- Linkage Design Features

The functional block diagram of the electronics is shown in figure 4. The electronics effort focussed on the development of an analog circuit interface board to drive the sensors, condition their output for the off-the-shelf A/D board, and perform certain additional tasks required during assembly of the devices. In addition, we developed software to collect the data from the A/D board, convert it into human joint angles, display the results, and transmit the joint angles through a serial port with an update rate of 100 Hz. The key element of the software is the hand calibration routine, which takes input from the user, combines it with fixed values from the sensor calibration and linkage geometries, and produces a set of look-up tables which map the sensor readings directly into human finger angles.

This combination of features allows an individual, regardless of hand size or shape, to place the device on their hand, perform the calibration, and get actual finger angles as a direct output. The calibration tables may be stored to allow the same individual to use the device again with a minimum amount of reconfiguration.

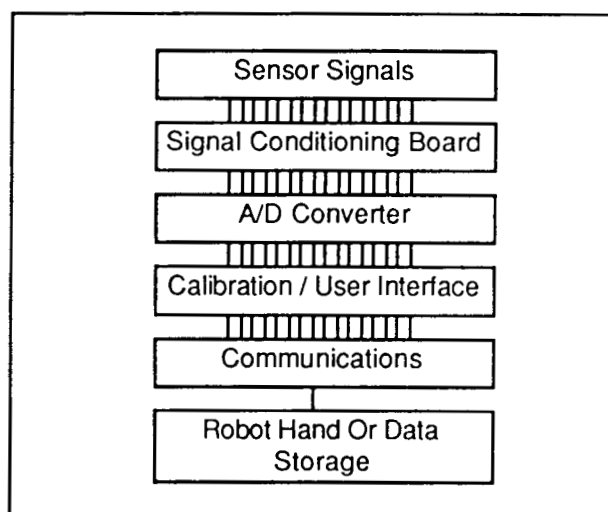


Figure 4- Electronics Block Diagram

SYSTEM DESCRIPTION:

The resulting system illustrated in figure 1 incorporates the following components:

- Exoskeleton - the mechanical structure which attaches the sensors to the human hand,
- Hall Effect sensors - 16 sensors, 4 per digit, 3 measure bending, 1 measures medial / lateral motion,
- Connection board - supplies power to the sensors, and returns their signals to the signal conditioning board,
- Signal conditioning board - conditions the signals for input to the A/D board,
- 12 bit, 16 channel A/D converter - creates digital representation of the analog sensor outputs,
- AT compatible micro-computer - provides data storage, date transmission, and calibration capabilities,
- RS422 serial port - transmits data at 19,200 baud for input to other computer systems, and

Proprietary software package - provides calibration, display, and communications functions.

CONCLUSIONS:

The Dexterous Hand Master system will allow users of dexterous robots to fully and rapidly understand which types of tasks can usefully be performed by these robots. It will also allow detailed study of the kinematics of the human hand, and the level of dexterity required to perform different tasks, which can enhance robotic applications. To date, four systems have been ordered, with three delivered. In the future we look anticipate developing more advanced versions of this technology featuring faster calibration routines, measurement of wrist joint angles and force reflection.

REFERENCES:

1. Jacobsen, S.C., Iversen, E.K., Knutti, D.F., Johnson, R.T., & Biggers, K.B., "Design of the UTAH/MIT Dexterous Hand," Proceedings of the IEEE 1986 International Conference on Robotic and Automation, San Francisco, California.
2. Jacobsen, S.C., Wood J.E., Knutti, D.F., & Biggers, K.B., "The UTAH/MIT Dexterous Hand: Work in Progress," International Journal of Robotics Research, Vol. 3, No. 4, Winter 1984, pp21-50.

APPLICATION OF MODEL BASED CONTROL TO ROBOTIC MANIPULATORS

Lyman J. Petrosky, Senior Engineer
Robotics and Mechanical Design
Westinghouse Advanced Energy Systems
Madison, PA 15663

Irving J. Oppenheim, Assoc. Prof.
Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, Pennsylvania, 15213

ABSTRACT

A robot that can duplicate human motion capabilities in such activities as balancing, reaching, lifting, and moving has been built and tested at Carnegie-Mellon University. These capabilities are achieved through the use of real time Model-Based Control (MBC) techniques which have recently been demonstrated. MBC accounts for all manipulator inertial forces and provides stable manipulator motion control even at high speeds. To effectively demonstrate the unique capabilities of MBC, an experimental robotic manipulator was constructed which stands upright, balancing on a two wheel base. The mathematical modeling of dynamics inherent in MBC permit the control system to perform functions that are impossible with conventional non-model based methods. These capabilities include:

- Stable control at all speeds of operation;
- Operations requiring dynamic stability such as balancing;
- Detection and monitoring of applied forces without the use of load sensors;
- Manipulator "safing" via detection of abnormal loads.

The full potential of MBC has yet to be realized. The experiments performed for this research are only an indication of the potential applications. MBC has no inherent stability limitations and its range of applicability is limited only by the attainable sampling rate, modeling accuracy, and sensor resolution. Manipulators could be designed to operate at the highest speed mechanically attainable without being limited by control inadequacies. Manipulators capable of operating many times faster than current machines would certainly increase productivity for many tasks.

INTRODUCTION

The design of a control system for manipulators is a formidable task due to the complexity of the nonlinear coupled dynamics. The goal is the calculation of actuator torques which will cause the manipulator to follow any desired trajectory. Research has produced many manipulator control schemes ranging from simple Proportional + Integral + Derivative (PID) control to nonlinear feedback control. In a broad sense, two basic categories of control design are found in the literature. The first contains the robust control methods in which the control is able to overpower the system's nonlinear coupled dynamics. The second contains the *model-based* control (MBC) methods in which many of the system nonlinearities are calculated via a system model and the nonlinear system forces are canceled by the actuation forces. For this research we have chosen to use the *Computed-torque model-based* control described in the next section.

Recent advances in computational hardware have made it possible to evaluate in real time the equations of motion of robotic manipulators. Khosla¹ at Carnegie Mellon University (CMU) was the first to demonstrate the feasibility of real time MBC using an inexpensive computer system for control of a six degree of freedom manipulator, the CMU Direct Drive Arm II. The current work builds on this accomplishment and explores additional manipulator capabilities that the existence of real time MBC creates.

PRECEDING PAGE BLANK NOT FILMED

The assumptions required to apply the methods presented here are:

- System is amenable to mathematical modeling
- Suitable control law can be formulated
- Mathematical model and control law can be evaluated in real time
- Necessary physical variables can be instrumented

CONTROL APPROACH

Computed-Torque Control

*Computed-torque*² control is a *model-based* control scheme which strives to use the complete dynamic model of a manipulator to achieve dynamic decoupling of all the joints using nonlinear feedback. The dynamic model of the manipulator is described by the Lagrangian derived equations of motion:

$$\sum_{j=1}^N D_{ij} \ddot{q}_j + \sum_{j=1}^N \sum_{k=1}^N C_{jk}(i) \dot{q}_j \dot{q}_k + g_i = \tau_i \quad \text{for } i = 1, \dots, N. \quad (1)$$

where the q are the joint coordinates. The τ_i are the externally applied joint actuation torques/forces. The inertial D_{ij} , centrifugal and Coriolis $C_{jk}(i)$, and gravitational g_i coefficients of the closed-form dynamic robot model in Equation 1 are functions of the instantaneous joint positions q_i and the constant kinematic, dynamic and gravity manipulator parameters. The kinetic energy gives rise to the inertial and centrifugal and Coriolis torques/forces, while the potential energy leads to the gravitational torques/forces. Actuator dynamics can be incorporated in the dynamic robot model by additions to the Lagrangian energy function.

The *Computed-torque* algorithm begins with a calculation of the required torque to be applied to each of the joints (in vector notation):

$$\begin{aligned} \tau &= \tilde{D}u + \tilde{H} + \tilde{g} \\ \tilde{H}_i &= \dot{q}^T \tilde{C}(i) \dot{q} \end{aligned} \quad (2)$$

where u is the commanded joint accelerations. The "~" indicates that these matrices are calculated from the estimated system parameters. The resulting dynamic equations for the closed-loop system are:

$$\ddot{q} = u - D^{-1} \{ (D - \tilde{D})u + (H - \tilde{H}) + (g - \tilde{g}) \} \quad (3)$$

If the system dynamic parameters are known exactly, then $\tilde{D} = D$, $\tilde{H} = H$, and $\tilde{g} = g$, then the closed loop system is described by:

$$\ddot{q} = u \quad (4)$$

which is the equation for a set of decoupled second order integrators. The commanded acceleration u_i is then formulated to incorporate the error feedback signal and the reference signal. After decoupling, each joint acts as a second order integrator, therefore the control law is given the form:

$$u_i = \ddot{q}_{id} - 2\zeta\omega(\dot{q}_i - \dot{q}_{id}) - \omega^2(q_i - q_{id}) \quad (5)$$

which causes each joint to act as a second order damped oscillator with natural frequency ω and damping ratio ζ . The form of the equation causes the joint to track the desired joint values q_{id} , \dot{q}_{id} , and \ddot{q}_{id} .

The computed-torque control defined above is based on the assumptions that the system model is accurate and that all joints are actuated. For this research the dynamic parameters of the experimental manipulator were manually measured to provide an accurate system model. We assume in all simulations that the dynamic model is accurate. The second assumption, that all the joints are actuated, does not apply for the experimental system which requires dynamic balancing. However, a suitable control law exists based on the work of Petrosky³. The method, called *hierarchical partitioning*, is directly applicable to the balancing problem, is robust, and gives an intuitive feel for the system's behavior. For the experimental balancing manipulator, the control law, which treats the manipulator as a single inverted pendulum, is merged with the computed-torque control to complete the algorithm.

Determination of Applied Forces

Indirect determination of applied forces (*i.e.* without the use of load sensors) is accomplished by comparison of the manipulator mathematical model and the observed manipulator behavior. A simple example of this is the algorithm for payload determination for the balancing manipulator. Payload estimation can be performed for a balancing manipulator without accurate knowledge of the actuation forces and accelerations, and the required calculation can be performed on-line in real time. Consider the equation of motion for pivoting about the base of the dynamically balanced manipulator:

$$\tau_i = \sum_{j=1}^N D_{ij} \ddot{q}_j + \sum_{j=1}^N \sum_{k=1}^N C_{jk}(i) \dot{q}_j \dot{q}_k + g_i \quad \text{for } i = \text{base joint} \quad (6)$$

The base joint of a balanced manipulator is not actuated, therefore $\tau_i = 0$. However, if the payload value is incorrect, then this equation will evaluate to a non-zero value of τ_i when the observed values of the joint variables are entered. The difference indicates the value of the payload which is given by:

$$\Delta P = - \left(\frac{\partial \tau_i}{\partial P} \right)^{-1} \tau_i \quad (7)$$

where ΔP is the difference between the actual payload and the current estimated value. Under ideal conditions this equation would yield the correct payload value in a single sample; however, the accuracy of the values for \ddot{q}_j can be exceedingly poor if they obtained by double differentiation of position measurements. This was the case in the experimental system, but the problem was overcome by the use of a parameter estimator.

EXPERIMENTAL SYSTEM

Manipulator and Sensors

The experimental manipulator is supported solely on two wheels. It is a double inverted pendulum operating in a plane, and constantly requires active balance motions to prevent falling. The manipulator consists of the servo driven wheeled base, a lower arm section, an elbow joint with drive servo, an upper arm, and an electro-magnet gripper at its tip. It is constructed primarily of aluminum and has a total weight of 13 kg. The two wheels mounted on a single shaft provide effective out-of-plane stability. The tip of the manipulator can reach to a height of 1.8 meters when fully extended, and can be lowered to touch the floor.

The manipulator wheels and elbow joint are each driven by Aerotek servos rated at 1.3 N-m peak torque. The elbow joint has a chain reduction ratio of 57.6:1 and the drive wheels have a chain reduction of 4.8:1. The chain reduced servo arrangement was chosen over direct drive to save weight, and over gear-reduced or harmonic drive to mitigate costly damage in the event of a severe floor collision.

The sensors utilized for manipulator control are:

- Inclination RVDT - a rotary differential transformer measures the angle between the floor surface (via a feeler) and the lower arm.
- Motor Encoders - each servo has an optical encoder of 500 counts per revolution which runs a hardware counter read by the parallel interface board.

Computer Control System

The computer control system hardware consists of a Motorola M68000 based single board computer as the master CPU, a Marince Array Processor Board (APB), an Analog to Digital Converter (ADC) 32 channel input board, a Digital to Analog (DAC) 4 channel output board, a 96 line Parallel Input/Output (PIO) Interface board, and a CRT terminal. The master CPU drives the bus communications, terminal interface, and an interface to a VAX computer. The VAX computer serves as the disk storage for the system programs and as the post processor of the experimental data.

The Marince array processor is a high speed programmable single board processor with an instruction cycle of 125 ns. It is used to perform the calculation intensive operations required to implement MBC. The board has fixed point multiplier and addition hardware which are used for floating point operations. The floating point addition or multiplication routines execute in approximately 1 μ s. Negation requires 125 ns. Computation of the sine/cosine pair requires 15 μ s. Additional routines perform data type conversion and other functions required to format the sensor data.

Processing for *real time control* is done by the Marince processor exclusively. Manipulator trajectory calculations are handled by the M68000 CPU on a time sharing basis. In operation a timer interrupts the CPU at each sampling instant. The CPU copies the sensor data to the Marince array processor memory, and initiates Marince execution. The Marince formats the data, does scaling operations, performs the trigonometric functions, and then calculates the inverse dynamics. The formatted output data is ready in less than 0.5 ms. The Marince returns to control data to the CPU which outputs it to the DAC's. The cycle time is fast enough that the control algorithm and dynamic model can be evaluated at a sampling frequency in excess of 1000 Hz.

EXPERIMENTAL RESULTS

The experimental manipulator was fully reliable in maintaining balance for long periods while performing a variety of tasks. The base moves approximately ± 3 mm to maintain balance and the tilt varies by ± 0.0063 rad. This motion does not indicate a flaw in the balancing algorithm, but rather the motion results from being at the limit of tilt resolution of the RVDT sensor used with the floor feeler; the RVDT signal variation corresponds to the magnitude of a single digital count. Because the base dimension of the experimental system is zero, it is physically impossible for the manipulator to balance without some on going motion.

The manipulator proved very resistant to upset; its recovery ability appears to exceed that of a human under similar magnitude disturbances. Figure 1 records the transient response of the manipulator to a severe impact (base position q_1 and tilt q_2). The manipulator moved forward to balance and then quickly returned to the original position. The manipulator was also forgiving (compliant) of collision. The manipulator would bounce lightly off an obstacle and come to rest simply leaning against it. When commanded to back away from the obstacle, the manipulator would resume balancing as soon as contact was broken.

Figure 2 records the transient response of the manipulator under two cycles of application of a payload of 0.811 kg, suddenly added or removed. The payload compensation algorithm quickly determines the new payload and adapts the control, restoring the manipulator to its original position. It was possible to carry large payloads with the experimental manipulator. The manipulator used its own weight to balance and transport payloads ranging up to 3.2 kg, which is 25% of its system weight, without difficulty. The results of payload estimation in a non-transient condition show a noise level of only ± 26 gm which is 0.2% of the system mass.

Another payload experiment successfully demonstrated the development and control of lateral force through the motion of system masses. A rope attached the manipulator to a heavy mass on a horizontal surface, and the manipulator was used to pull the mass across the table, against the force of friction, for some target distance. The manipulator developed a lateral force through the movement of its center of mass to a point behind its wheel axis, producing a resulting force in the rope; the system displays balance through the movement which ensues when the lateral force reaches and exceeds the friction force. The manipulator thereby demonstrates the approach used by a human to pull a heavy weight across a floor.

CONCLUSIONS

The feasibility of utilizing real time Model Based Control (MBC) for robotic manipulators has been demonstrated. The experimental results demonstrate the effectiveness of the control approach and of the payload estimation/adaptation algorithm developed for this effort. The mathematical modeling of dynamics inherent in MBC permit the control system to perform functions that are impossible with conventional non-model based methods. These capabilities include:

- Stable control at all speeds of operation;
- Operations requiring dynamic stability such as balancing;
- Detection and monitoring of applied forces without the use of load sensors;
- Manipulator "safing" via detection of abnormal loads.

This work demonstrates an initial implementation of the above features for a robotic manipulator.

The full potential of MBC has yet to be realized and much work remains to be done. The experiments performed for this research are only an indication of the potential applications. Unlike conventional PID control, MBC is a theoretically complete control algorithm with no inherent stability limitations. Its range of applicability is limited only by the attainable sampling rate, modeling accuracy, and sensor resolution. Manipulators could be designed to operate at the highest speed mechanically attainable without being limited by control inadequacies. Manipulators capable of operating many times faster than current machines would certainly increase productivity for many tasks. These manipulators could also have build in safing in response to abnormal loading conditions.

ACKNOWLEDGMENTS

The research¹ was performed for the Department of Energy, Advanced Reactor and Nuclear System Technology Support, Program NE-85-001. We are indebted to the Department of Energy for sponsorship of this research under contract DE-AC02-85NE37947, *Dynamic Stability for Robot Vertical Reach and Payload*, and to Clint Bastin of DOE for his particular interest and support. We are also most grateful to Westinghouse AES for their cooperation in accommodating the residence of Mr. Petrosky at Carnegie-Mellon. Finally, we wish to acknowledge the contributions of Professor I. Shimoyama, Professor J. Bielak, and graduate student Eric Hoffman.

REFERENCES

1. Khosla, P.K., *Real-Time Control and Identification of Direct-Drive Manipulators*, PhD dissertation, Carnegie-Mellon University, 1986.
2. Markiewicz, B.R., "Analysis of the Computed-Torque Drive Method and Comparison with the Conventional Position Servo for a Computer-Controlled Manipulator", Technical Memorandum 33-601, Jet Propulsion Laboratory, Pasadena, CA, March 1973.
3. Petrosky, L.J., "Problem Substructuring Applied to the Design of a Controller for the Stabilization of a Double Inverted Pendulum", Master's thesis, Carnegie-Mellon University, 1986.

¹Disclaimer: The view, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official U. S. Department of Energy or Carnegie-Mellon University position, policy or decision, unless designated by other documentation.

PATENT STATUS: This document copy, since it is transmitted in advance of patent clearance, is made available in confidence solely for use in performance of work under contracts with the United States Department of Energy. This document is not to be published nor its contents otherwise disseminated or used for purposes other than that specified, before patent approval for such release or use has been secured upon request from the Patent Counsel, U.S. Department of Energy.

ORIGINAL PAGE IS
OF POOR QUALITY

7

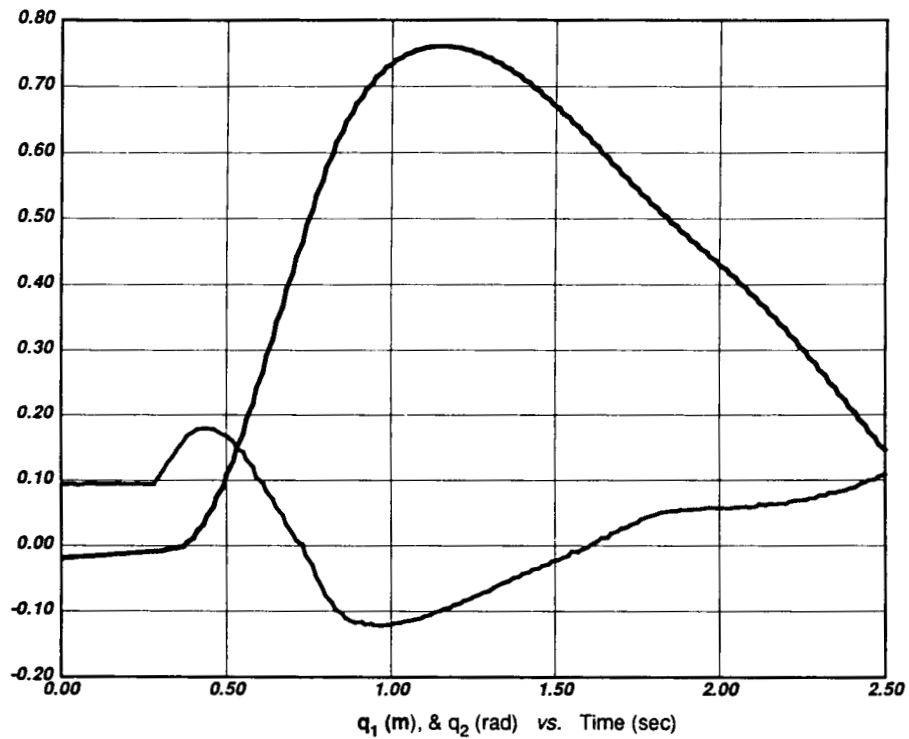


Figure 1: Response to a Lateral Impulse Load at 0.3 Seconds

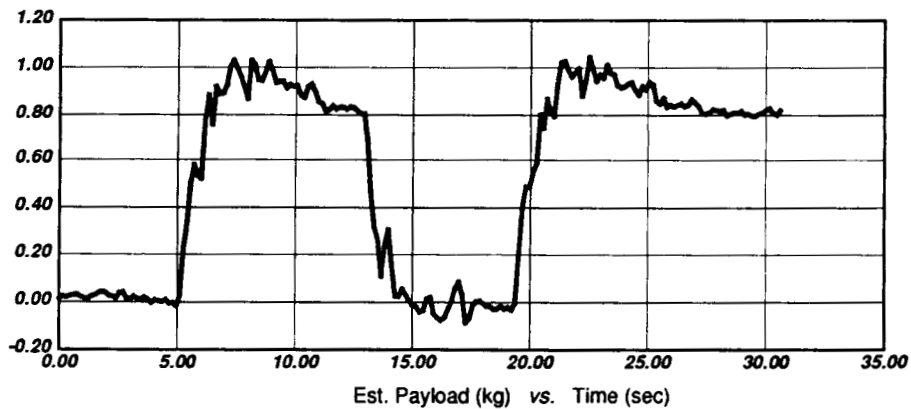


Figure 2: Payload of 0.81 kg Applied On/Off/On

DESIGN GUIDELINES FOR REMOTELY MAINTAINABLE EQUIPMENT

Margaret M. Clarke, Ph.D.
 Davoud Manouchehri
 Rockwell International
 12214 Lakewood Boulevard
 Downey, CA 90241
 (213) 922-0785

1. INTRODUCTION

The quantity and complexity of on-orbit assets will increase significantly over the next decade. Maintaining and servicing these costly assets represent a difficult challenge. Three general methods are proposed to maintain equipment while it is still in orbit. First, an extravehicular activity (EVA) crew can perform the task in an unpressurized maintenance area outside any space vehicle. Second, intravehicular activity (IVA) crew can perform the maintenance in a shirt sleeve environment, perhaps at a special maintenance work station in a space vehicle. Third, a telerobotic manipulator can perform the maintenance in an unpressurized maintenance area at a distance from the crew (who may be EVA, IVA, or on the ground). However, crew EVA may not always be possible; the crew may have other demands on their time that take precedence. In addition, the orbit of the tasks themselves may be impossible for crew entry. Also crew IVA may not always be possible as an option for equipment maintenance. For example, the equipment may be too large to fit through the vehicle airlock. Therefore, in some circumstances, the third option, telerobotic manipulation, may be the only feasible option. Telerobotic manipulation has, therefore, an important role for on-orbit maintenance. It is not only used for the reasons outlined above, but used also in some cases, that may act as backup to the EVA crew in an orbit which they can reach.

If equipment is to be serviced by a telerobotic manipulator, then the orbital replacement units (ORU's), which make up this equipment, must have a compatible interface with the telerobot. If EVA crew maintain the same piece of equipment at times and changeout the same ORU's, then the ORU's must also have a compatible interface with crew EVA suit limitations and capabilities. Rockwell is very aware of the necessity for interface compatibility between ORU's and their mode of maintenance (telerobot and/or EVA crew). The Space Transportation Systems Division (STSD) has, therefore, a continuing project to develop guidelines for ORU's to ensure their interface compatibility (Figure 1). This paper describes the work performed so far on ORU/telerobot interface compatibility.

STSD's work is progressing in three phases: telerobotic definition, ORU interface guidelines, and feasibility demonstration. Each phase is discussed below followed by a summary of the benefits resulting from this approach.

2. TELEROBOTIC DEFINITION

Rockwell STSD has already completed a project to define and describe a telerobotic manipulator arm, (the Extravehicular Teleoperator Assist Robot [ETAR]), capable of changing out common ORU's on present and future on-orbit equipment. The force reflecting arm (Figure 2), which features 7-degrees of freedom, was described in detail at the 1987 SOAR Conference. (Reference 1) This effort led to the remaining two phases of work.

3. ORU INTERFACE GUIDELINES

ORU interface guidelines were generated in a three-stage process: ORU identification, interface data base and requirements generation, and guidelines identification.

3.1 ORU Identification

The goal of this step was to identify specific ORU's on present and future planned satellites, manned space vehicles, and other on-orbit equipment. For example, 27 representative ORU's on satellites and scientific experiments were identified and analyzed. (Reference 2) In addition, over 1,000 Space Station ORU's were identified during Rockwell's Phase B Space Station Activity. These ORU's were classified into nine major types shown in Figure 3.

3.2 Interface Data Base and Requirements

The goal of this step was to compile a detailed data base of information on the ORU's identified in Step 3.1. Emphasis was placed on information that would impact the ORU's interface with a telerobotic manipulator and with EVA compatibility as a backup.

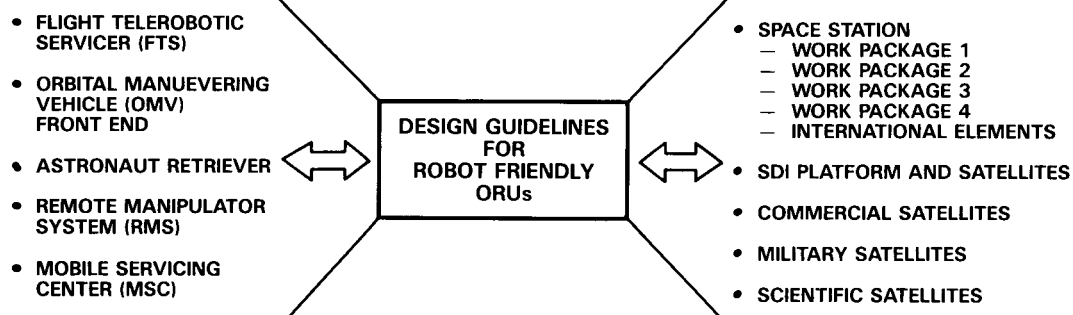


Figure 1. Design Guidelines Will Ensure Compatibility Between ORU's and the Robot

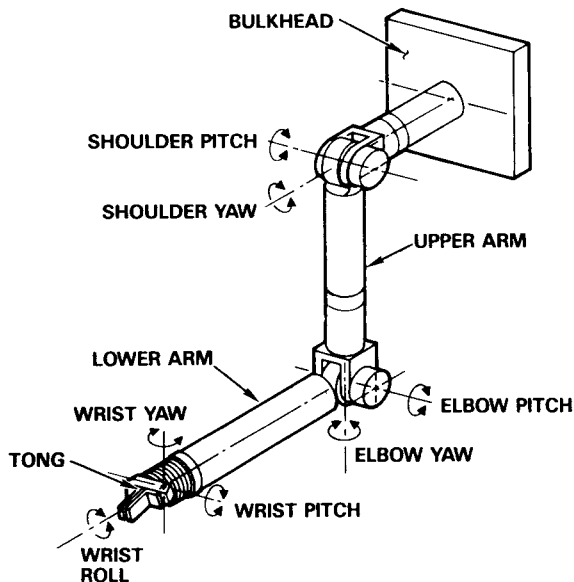


Figure 2. ETAR Arm Concept

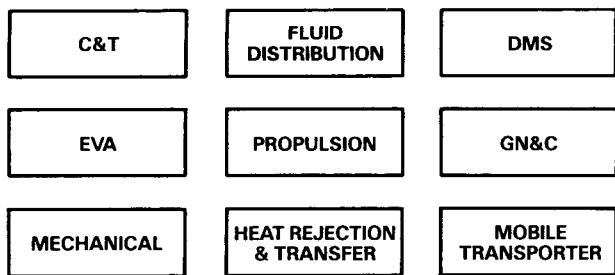


Figure 3. Identified ORU Types

We relied on a wide variety of sources for information including NASA engineers. Other data were obtained from two former astronauts who had performed EVA maintenance tasks, from vendors of commercially available space rated fasteners and connectors, and from other industries that use modular maintenance concepts e.g., commercial airlines. In addition, Rockwell ORU design engineers were asked to fill out an ORU Interface Requirements Questionnaire giving as specific information as they could on the ORU's mass, volume, shape, dynamics and kinematics of the changeout, cold plate contact, and so on. While the information was of necessity at a high-level and preliminary, it was very useful in allowing us to classify the large numbers of ORU's in terms of their requirements for interface with a teleoperated manipulator. An example of a possible classification scheme is shown in Table 1.

3.3 Guideline Identification

By using the data base generated in Step 3.2, we then identified specific guidelines for the various classification of ORU interfaces. The guidelines met the requirements for compatibility with a telerobotic manipulator with EVA compatibility as a backup. For example, a small number of fluid connectors, electric connectors, and mechanical fasteners were selected as having the potential to provide compatible interface with over 90 percent of the ORU's in our data base. Figure 4 shows an example of a candidate mechanical fastener, the over center clamp.

Table 1. Examples of Possible Interface Requirements

Mechanical fasteners
• High load bearing
• Low load bearing
• Special tools needed
Fluid Connectors
• Line Size
— Type of fluid
• Pressure
Electronic Connector
• Power
— Voltage
• Data Rate
• Pin Sizes

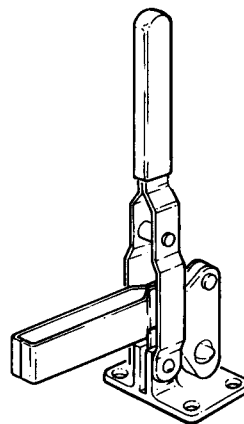


Figure 4. Over Center Clamp

4. FEASIBILITY DEMONSTRATION

To demonstrate the feasibility of the proposed guidelines, several ORU changeout demonstrations were conducted in the Rockwell Automation and Robotics Facility (Figure 5).

The facility contains an electromechanical teleoperated manipulator with two 7-degrees of freedom slave arms driven by a replica master. The facility also contains a four-degrees of freedom transporter to move the slave through its work place. Cameras are onboard the slave and also fixed at other locations in the work place.

Task boards contain mockups of a large variety of ORU's. A mockup of a standard data processor black box ORU was built in accordance with design and performance specifications of the Space Station data processing system. It was compatible with both EVA and telerobotics ORU design standards (Figure 6). The standard data processor (SDP) slides in position (along a rack) and is guided by a built-in key design. The electrical and fiber optic connectors are all blind mated and self aligned. These connectors are located in the back of the unit. The SDP can be secured in position by a simple forward motion of a handle bar that uses an EVA hand-hold design. This handle bar is designed as part of the rack and generates enough force to ensure proper contact with the cold plate located under the SDP.

In order to evaluate this concept, a series of tests were conducted in the Rockwell Automation and Robotics Facility. In all cases there

A861110-V20

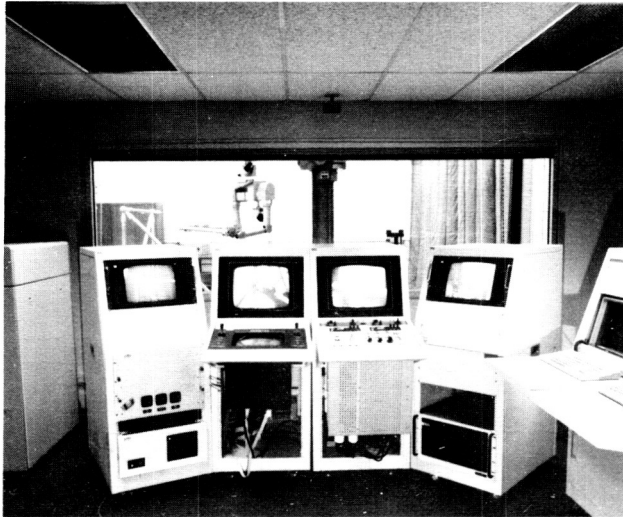


Figure 5. Rockwell Automation and Robotics Facility

was no direct visual contact between the operator and the work site. The operator had access to six television camera views—two of which were located on the slave arms and four of which provided views of the work space at four different angles. The force and torque feedback sensors did not decrease the operation time nor enhance the operation performance. Therefore, in most test cases, the force and torque sensors were turned off. Simple parallel jaws with friction pads were used as end effectors and appeared to be fully compatible with the SDP handle bars. Overall, these tests confirmed the simplicity of the SDP replacement operations. They also indicated that the developed guidelines provided compatibility between the robot and ORU's.

5. BENEFITS

Numerous benefits may be realized by our approach in suggesting standardization of connectors between many ORU's. Costs for design, development, test, and evaluation (DDT&E) of connectors and racks, as well as crew training, are reduced because the number of different types is reduced. Fewer spares must be warehoused. In addition, fewer varieties of tools and end effectors are required. Also, capability to reconfigure is increased. Furthermore, future automation becomes more efficient because standardized end effectors are used. Consequently, less robotic software must be written.

A870514C-7c

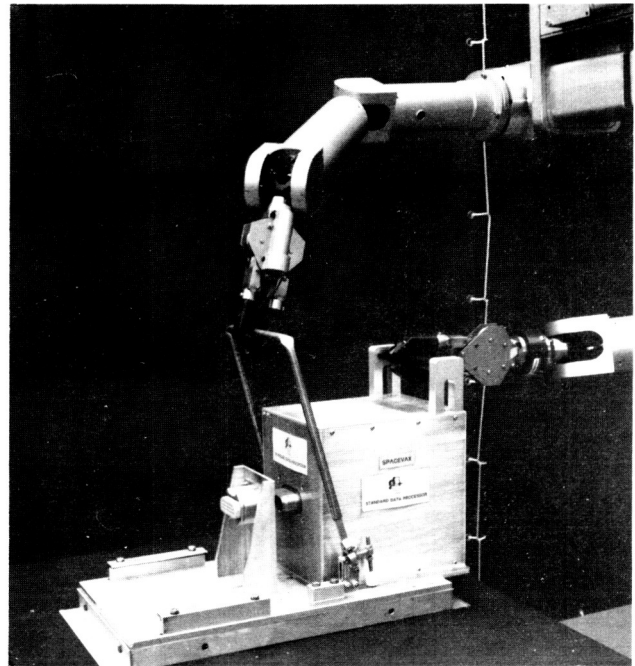


Figure 6. RM-10A Manipulator Arms Have Demonstrated SDP Changeout

Finally, these benefits would also assist the whole Space Station integration effort in that fasteners could be common across all four work packages.

6. REFERENCES

1. Clarke, Margaret M., Divona, Charles J., and Thompson, William M., Manipulator Arm Design for the Extravehicular Teleoperator Assist Robot (ETAR): Applications on the Space Station, Proceedings, 1987, First Annual Workshop on Space Operations Automation and Robotics (SOAR), Houston, Texas, August 1987, Pages 471-475.
2. Clarke, M.M., Thompson, W.M., Divona, C.J., "Requirements and Conceptual Design of the Manipulator System for the Extravehicular Teleoperator Assist Robot (ETAR)", SSS 86-0139, Rockwell International, Space Station Systems Division, Downey, California, November 1986.

EXPERT SYSTEMS FOR STRUCTURAL ANALYSIS
AND DESIGN OPTIMIZATION

Hasan Kamil
Ashok K. Vaishl
Structural Analysis Technologies

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

POWER SYSTEM AUTONOMY DEMONSTRATION FOR SPACE STATION

Gale R. Sundberg
Karl A. Faymon
NASA Lewis Research Center

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

AN EXPERT SYSTEM FOR RESTRUCTURABLE CONTROL

Jonathan Litt
Propulsion Directorate
U.S. Army Aviation Research and Technology Activity-AVSCOM
Lewis Research Center
Cleveland, OH 44135

ABSTRACT

Work in progress on an expert system which restructures and tunes control systems on-line in real-time is presented. The expert system coordinates the different methods involved in redesigning and implementing the control strategies due to plant changes.

INTRODUCTION

A restructurable control system has the ability to redesign itself on-line to compensate for a significant change in the plant. Restructurability is a valuable feature because it allows a closed-loop system to continue operating in an acceptable manner even in the face of major changes to the plant. Examples of plants with major changes are aircraft with battle damage or engines with foreign object damage. With an invariant control system designed for the original plant, an aircraft that experienced battle damage may now only be able to limp home. In the worst case it would be unstable and crash. With a redesigned control system for the new, altered plant, the plane may be able to carry out all or part of its mission with only slightly reduced capabilities and it is more likely that it will return safely.

Restructurable control is applicable to mechanical problems such as actuator or control surface failures. Most of the redesign strategies in the literature work by redistributing the forces and moments of the failed actuator or missing surface over the remaining components to compensate for the lost components. The research by Looze, et al has concentrated on a linear quadratic approach to the redesign procedure [1]. Horowitz, et al have applied quantitative feedback theory to

control system reconfiguration [2]. Raza and Silverthorn have used the pseudoinverse of the control matrix and generalized input vectors to achieve the desired responses along orthogonal axes [3]. The technique in [3] is similar to the control mixer concept for reconfiguration described by Rattan [4].

The goal of this paper is to describe a way to tie together some of the previous work in the field so as to achieve a highly survivable control system. A highly survivable system can successfully restructure in response to a multitude of different failures. In general, previous restructurable controllers have been specifically designed for a single failure type. Each design method used is valid for its specific application. However, none is "optimal" nor even applicable in all situations. Thus, to achieve a highly survivable system, it is necessary to identify the current dynamic characteristics of the plant and to determine which of the possible solutions is the best in some sense under the given circumstances. To accomplish this decision making in an uncertain environment with potentially conflicting mission objectives, some type of intelligence will be required. Hence the concept of an expert system to coordinate the different redesign strategies is proposed.

An expert system consists of three independent parts: an inference engine, a rule base, and a knowledge base. The rule base is a set of heuristics or rules-of-thumb which apply to the type of problem at hand. The knowledge base is a collection of information specific to the current situation. The inference engine is a program which applies the rules to the knowledge base in order to glean new information or to determine if an assumption is justified. When new

information is asserted, it is stored in the knowledge base.

BACKGROUND

The idea of restructurable control has appeared recently, mainly with respect to aircraft. Battle damage has been considered a perfect application for the research. Commercial airliners are also a possible vehicle for the work. Several accidents and near accidents where the pilot was able to recover and land the plane after analyzing the problem have been discussed in relation to restructurable control [5].

Thus this strategy is very attractive for both civilian and military aeronautics and propulsion applications. Creating the ability in a plane to restructure its control system after damage to continue at a level of performance similar to its original design specifications is highly desirable. It is also important to remember that the main ideas here are not limited to airplanes. They can be applied to a wide variety of systems with inherent redundancy.

EXPERT SYSTEM COMPONENTS

The proposed overall structure of the expert system is shown in Figure 1. It consists of (1) an inference engine, (2) a control system restructuring knowledge and rule base, and (3) a controller tuning knowledge and rule base. The control system restructurer is already partially implemented. In the future we plan to incorporate an on-line controller tuning expert system into the overall system. It will share the inference engine with the reconfiguration expert system.

An inference engine can work with any appropriately structured knowledge base and rule base; it is not linked in any way to the application. Likewise, a rule base can be used with any appropriate knowledge base.

The inference engine developed for this application is capable of performing symbolic and numerical calculations required to evaluate certain rules. It can also execute generalized rules with previously established facts from the knowledge base to infer new facts. In addition, it has the ability to perform what-if type reasoning by trying different scenarios if more than one is appropriate.

The knowledge base of the restructurable

control system consists of information about the plant and control systems. For a linear system such parameters as the system matrices and the original controller gains are stored. There are also specifications on the actuators such as linear ranges and characteristics. Information stored here can change in response to plant changes. It is changed or updated as new facts become available.

The rule base of the control system restructurer contains rules about control system design. These range from top-level control design methods to low-level details such as definitions of controllability and observability. The rules may contain numerical expressions to be evaluated (such as whether a realization is minimal) and may contain variables to be given values by the inference engine during the discovery of new facts.

A separate knowledge base will be required for the tuning system. It will contain response characteristics associated with a well-tuned loop of the type in question. It also will have data on the previous responses obtained in the tuning process.

A rule base for controller tuning will be created also. The heuristics will use the previous tuning and plant information to determine an appropriate tuning paradigm.

Figure 2 shows the interaction of the two expert systems with the overall system.

SYSTEM CAPABILITIES

Figure 3 shows the anticipated future setup of the overall system. It shows a hierarchy with an expert system receiving information from a system identifier and a pattern extractor. This information is used in the restructuring of the controller for the altered plant. In the current setup, the plant simulation, the controller, and the expert system are all written in compiled LISP running on an LMI Lambda LISP machine. The system identifier and the pattern extractor are not yet implemented. The simulation consists of a realization of a linearized system in the form of matrices (A,B,C,D) and the states are evolved using Euler integration. Presently the expert system uses a model of the plant directly from the simulation. The linear model is of the form

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

A change in the model prompts the expert system to analyze and redesign the control. The new controller replaces the old one in the simulation and the states continue to evolve.

The restructuring strategies that the expert system can currently use involve the pseudoinverse of B [3,4]. The expert system takes a realization (A,B,C) and manipulates it, using the Kalman Structure Theorem for instance, until it is minimal and B'B has full rank. If the expert system can achieve this goal, the equation

$$K = (B^T B)^{-1} B^T (A - (A_0 - B_0 K_0))$$

is used to determine the new controller matrix. Here A and B are the altered system matrices and $(A_0 - B_0 K_0)$ is the reduced order version of the closed-loop system matrix of the full order model.

Examples of the heuristics used in the situation described above are:

1. if (A,B,C) is controllable and observable
then realization is minimal
2. if B'B is full rank
then pseudoinverse of B exists
3. if (A,B,C) is not minimal and (A,B,C) is minimum phase
then find a minimal realization
4. if pseudoinverse of B exists and realization is minimal
then
 $K = (B^T B)^{-1} B^T (A - (A_0 - B_0 K_0))$

These rules, presented here in pseudo-code, are the type of heuristics contained in the rule base.

A user interface exists for use in the development stage. In a delivery system there will be no need for such an environment as the system will run without human intervention.

The expert system runs only when invoked, for example when the control needs to be redesigned. Currently, it is invoked by manually stopping the simulation and running the expert system. The simulation must then be restarted. This is necessary at present because the simulation and the expert system both run on the same processor and no system identification scheme has yet been implemented. In the future the identifier will communicate with the expert system and cause it to start redesigning when a significant change in the system matrices occur.

CONCLUDING REMARKS

The expert system is able to handle a variety of reconfiguration situations. For these cases, the new controller is designed and implemented in a matter of seconds. Naturally the redesign time depends on the order of the system.

At present a few of the control design algorithms from the literature have been implemented. More have to be included in addition to incorporating any other work, both new and existing, that is deemed necessary for the system to work well.

Some work has been done in the area of controller tuning by pattern recognition techniques for single-input single-output systems [6]. We intend to extend the methodology to multiple-input-multiple-output systems.

Currently the LISP machine is doing the numerical calculations. For the system to run in real time, the number crunching will have to be moved off the LISP processor to a numeric processor such as an array processor.

A system identifier will be implemented in the future. In the near term one might be implemented on the LISP machine. Eventually a microprocessor-based system identifier should be connected to the plant and signal the expert system if a significant change occurs in the model.

An on-line pattern extractor which will determine the response features will also have to be developed. These features will be passed to the knowledge base of the tuning expert system.

The simulation currently residing within the Lambda will be moved to an Applied Dynamics AD100 simulation computer. This will allow a full nonlinear simulation to be implemented and it will run in real time. When the interface between the two is completed, the capability will exist to test the expert system in a realistic situation.

REFERENCES

1. Looze, D. P., Weiss, J. L., Eterno, J. S., Barrett, N. M., "An Automatic Redesign Approach for Restructurable Control Systems," Proceedings of the IEEE National Aerospace and Electronics Conference 1985, Dayton, OH, 1985.

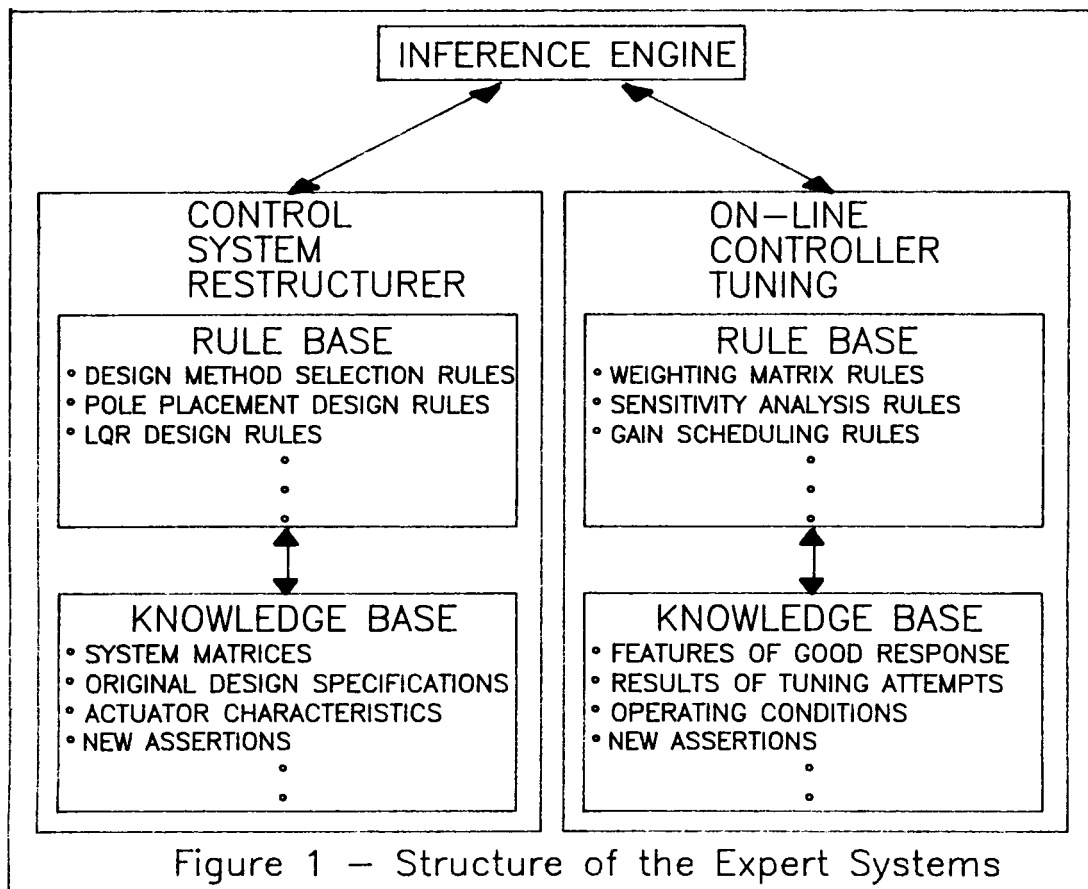
2. Horowitz, I., Arnold, P. B., Houpis, C. H., "YF16CCV Flight Control System Reconfiguration Design Using Quantitative Feedback Theory," Proceedings of the IEEE National Aerospace and Electronics Conference 1985, Dayton, OH, 1985.

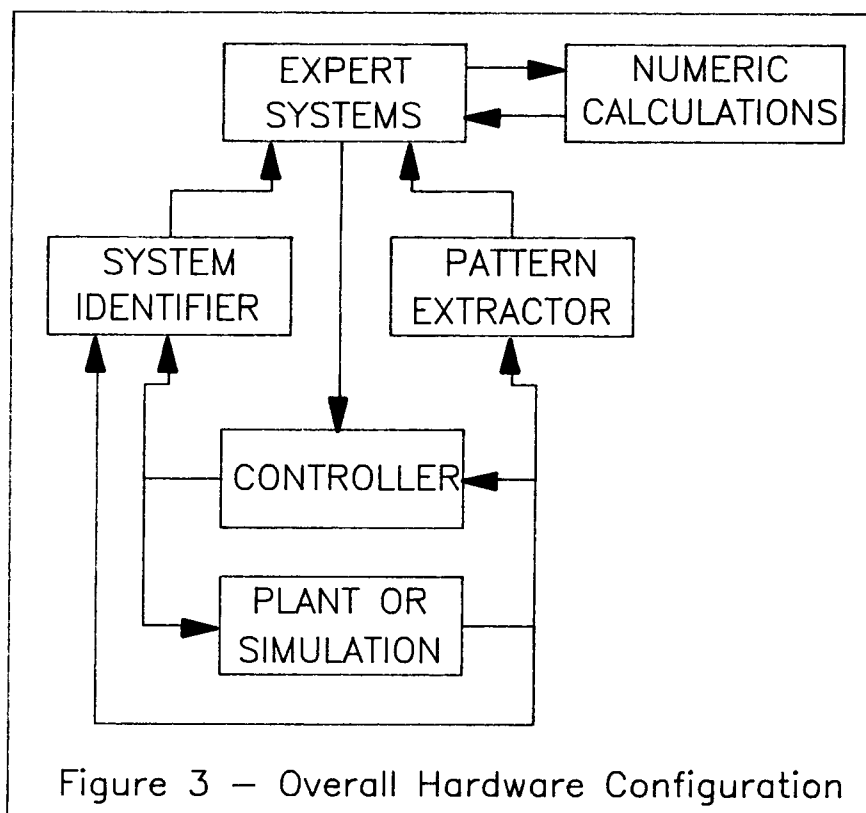
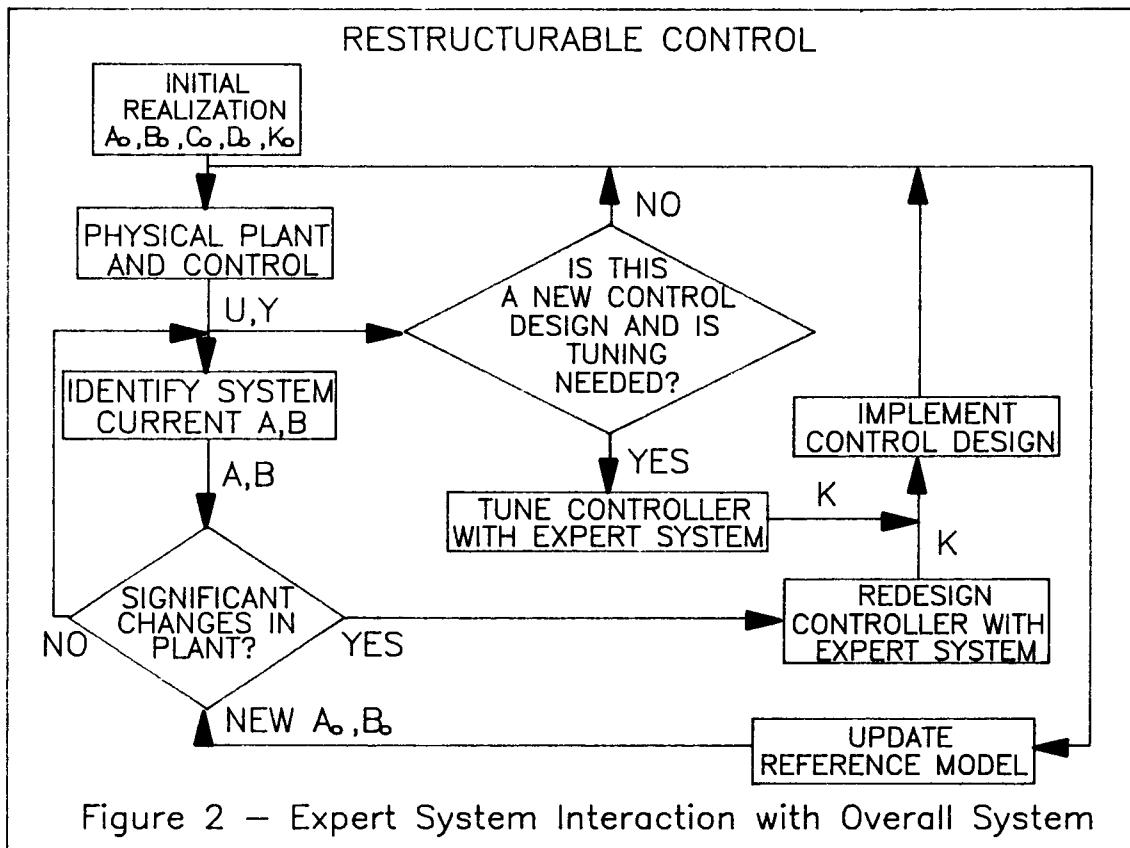
3. Raza, S. J., Silverthorn, J. T., "Use of the Pseudo-Inverse for Design of a Reconfigurable Flight Control System," Proceedings of the AIAA Guidance, Navigation and Control Conference, Snowmass, CO, 1985.

4. Rattan, K. S., "Evaluation of Control Mixer Concept for Reconfiguration of Flight Control System," Proceedings of the IEEE National Aerospace and Electronics Conference 1985, Dayton, OH, 1985.

5. Montoya, R. J., Howell, W. E., Bundick, W. T., Ostroff, A. J., Hueschen, R. M., Belcastro, C. M., "Restructurable Controls," NASA Conference Publication 2277, NASA Langley Research Center, 1982.

6. Litt, J., "An Expert System for Adaptive PID Tuning Based on Pattern Recognition Techniques," Proceedings of the ISA North Coast Conference, Cleveland, OH, 1986.





ADEPT: AN EXPERT SYSTEM FOR FINITE ELEMENT MODELING

Robert H. Holt
Autodesk, Incorporated

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

PHOTONIC (OPTICAL) PROCESSORS

David Jared

(Paper not provided by publication date.)

PRECEDING PAGE BLACK NOT FILMED

NEURAL NETWORKS

Marc Hosein

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

PACKAGING CONCEPTS FOR SUPERCHIP/VHSIC MODULES

Gloria Davis

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

EXPERT SYSTEMS APPLICATIONS FOR SPACE SHUTTLE PAYLOAD INTEGRATION AUTOMATION

by
Keith Morris

**Rockwell International
Space Transportation Systems Division
D282/900 FC43
12214 Lakewood Boulevard
Downey, California 90241
(213) 922-3700**

ABSTRACT

Expert systems technologies have been and are continuing to be applied to NASA's Space Shuttle orbiter payload integration problems to provide a level of automation previously unrealizable. The NASA's Space Shuttle orbiter was designed to be extremely flexible in its ability to accommodate many different types and combinations of satellites and experiments (payloads) within its payload bay. This flexibility results in different and unique engineering resource requirements for each of its payloads, creating recurring payload and cargo integration problems. Expert systems provide a successful solution for these recurring problems. The Orbiter Payload Bay Cabling Expert (EXCABL) was the first expert system, developed to solve the electrical services provisioning problem. A second expert system, EXMATCH, was developed to generate a list of the reusable installation drawings available for each EXCABL solution. These successes have proved the applicability of expert systems technologies to payload integration problems and consequently a third expert system is currently in work. This paper describes these three expert systems, the manner in which they resolve payload problems and how they will be integrated.

INTRODUCTION

Extreme flexibility, provided by the NASA Space Shuttle orbiter to accommodate diverse payload and cargo elements, makes payload and cargo planning, design, and integration a major activity. Each Shuttle mission carries a different set of payload elements, making the integration of these payloads into the orbiter payload bay a recurring complex planning, design, and installation problem. Expert systems technologies have been applied to these problems, providing automation to decrease these labor-intensive activities. The success of its first delivered expert-system-based automation tool inspired the Space Technology Systems Division (STSD) of Rockwell International to investigate the possibility of applying this technology to other payload planning, design, and integration problems. A second expert system was subsequently successfully delivered, confirming the value of expert systems in this problem domain. On the basis of these successes, a third expert system is currently being developed and additional payload integration problem areas are being examined for potential future expert systems applications.

THE PAYLOAD AND CARGO INTEGRATION AUTOMATION PROBLEM

The delivery of satellites and experiments into low earth orbit by the Space Shuttle involves many preflight engineering planning, design, and integration tasks. These tasks include selecting appropriate satellites and experiments to make up a mission payload set, locating each payload element within the payload bay, determining standard and unique services required by each payload, developing and documenting the payload to Space Shuttle orbiter interface requirements, selecting the individual cables necessary for providing the electrical services, preparing the electrical services cabling layout schematic, and preparing the technical instructions for mission payload installation and integration. A major goal for all payload planning, design, and integration tasks is to minimize the amount of change from mission to mission, thereby reducing paperwork, labor hours and turnaround time.

Complexity requires that payload integration planning and design tasks be carried out by teams of engineers. Each team uses both common and specialized engineering tools, some of which are extremely complex with rigid constraints. Any changes in planning and design methods have to take the use of these existing tools into consideration. The products of each team are used as initial planning and design data by one or more other teams. Likewise, the initial planning and design data for each team consist of the products of several other teams. Because of their interdependence, the products of these teams are integrated into a master mission plan and schedule.

Team technical support is supplied by one or more highly trained experts. These experts have many years of practical, as well as task related, experience. These task experts are rapidly reaching the age of retirement. Loss of an expert, regardless of cause, is an undesirable event not only with respect to the affected team's productivity, but also to the total payload integration design task productivity as a whole.

Real-world experience with space flight mission provisioning has shown that the ability to make mission manifest changes is mandatory. Certain other types of changes are to be anticipated because of further engineering analysis or design refinement. Changes caused by erroneous data or design omissions and errors are also to be expected.

Thus, change is a normal mode of operation and provisions must be made for change even close to launch time. Standardization and automation are two powerful methods used in the payload integration process to accommodate these changes.

To summarize, Space Shuttle payload and cargo integration planning and design tasks are a collection of iterative interrelated activities, supported by complex specialized tools, responsive to change, and led by vanishing experts. Automation, to be successful, must be tempered by these considerations. The following major objectives were established for each planning and design automation effort:

1. reduce engineering labor hours
2. retain technical expertise
3. reduce end-to-end process time
4. adapt to the existing operating techniques and environment

PAYLOAD BAY CABLING LAYOUT PLANNING AND DESIGN AUTOMATION

The Problem

The Space Shuttle payload bay cabling layout planning and design problem involves provision of the details required for the installation of cables to connect orbiter electrical services to the individual payload elements. Each Shuttle mission entails a different payload manifest, constituting a recurring planning and design problem. Mission payload manifest changes compound the problem further.

Standardized orbiter electrical services are provided through cables that connect the experiments and satellites to either the forward or aft orbiter payload bay bulkhead using standard mixed cargo harness (SMCH) panels. These cables are routed from the specific payloads to port and/or starboard standard interface panels (SIP's) and, from these SIP's, to covered cabling trays for further routing to either the forward or aft bulkhead (Figure 1). For efficiency, these

cables are provided from a standardized orbiter cable inventory.

Since these standardized cables must service all payloads, regardless of their location within the payload bay, they are almost always too long. The excess length of each cable must be dispositioned either by forming a foldback or loop (double foldback) within the routing tray. The trays are closed by covers that are located at designated locations along the tray. Cables with a diameter greater than 0.62 inch cannot be folded within the normal dimensions of the trays because of radius bend constraints. Therefore, a special height appending foldback cover is required to replace the normal tray cover at the location of such a fold. Also, at the point where the cable leaves the tray to be routed to the SIP or elsewhere, a special egress cover is required to replace the normal cover. Cables must also be separated by electromagnetic compatibility (EMC) class through special channels provided in the routing trays.

Cabling installation practices are also governed by numerous constraints and standard operating procedures. Based on heuristic knowledge, the above considerations, and the specific payload manifest, the cabling expert generates a hand drawn schematic that describes the cable routing solution. This schematic is subsequently used by a CAD/CAM specialist to produce a technical order (TO) schematic drawing of the cabling layout. An example of a typical TO schematic drawing is shown in Figure 2.

The Solution

The NASA Space Shuttle's payload bay cabling design task was the first automation problem that applied expert systems technology. An expert system, the Orbiter Payload Bay Cabling Expert (EXCABL), was completed in September of 1986 and has been in operational use since delivery. An overview of the EXCABL system is shown in Figure 3. The EXCABL system has completely automated payload bay cabling layout planning and design tasks. The cabling expert needs only to define the mission-unique payload requirements and constraints to generate the cabling solution CAD/CAM TO drawings and printed reports. This was facilitated by the initial construction of a mission-independent data base,

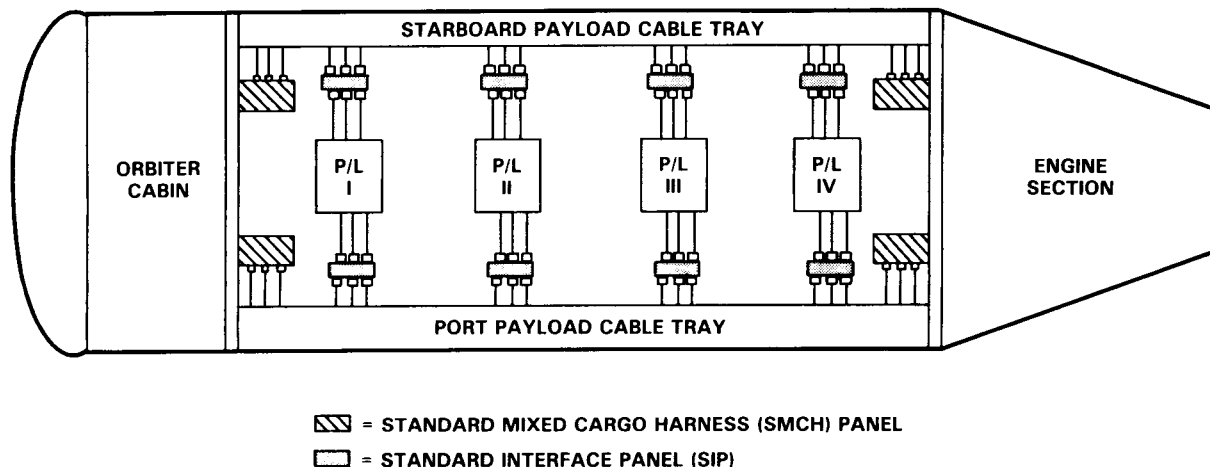


Figure 1. Shuttle Orbiter Payload Bay

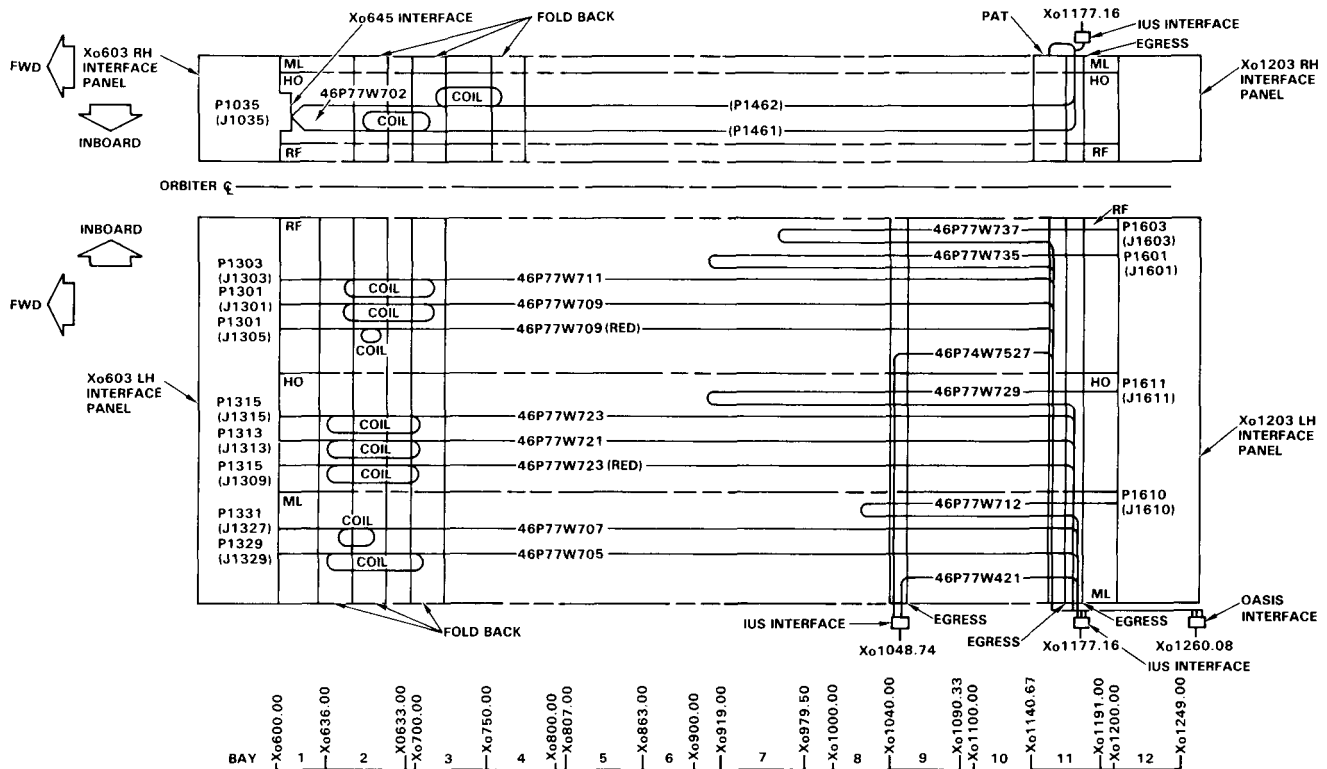


Figure 2. Typical EXCABL Drawing

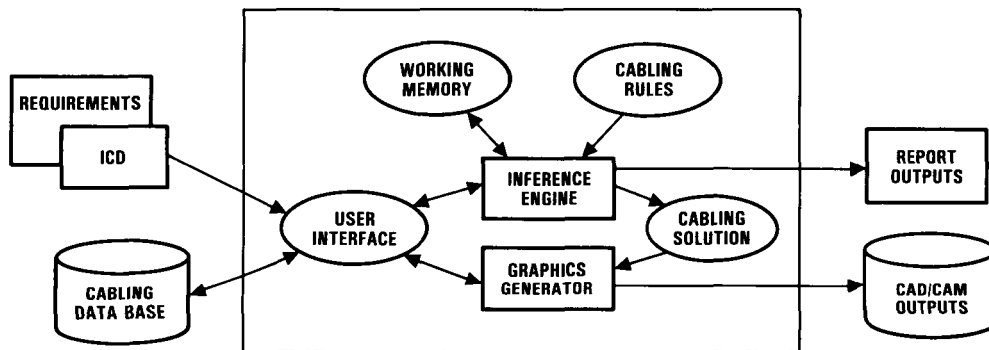


Figure 3. EXCABL System Overview

containing all of the necessary payload bay hardware information required to perform Space Shuttle's cabling. The cabling experts' previously required hand drawn cabling schematic is now automatically generated by the system and transferred into CAD/CAM inputs.

All major automation objectives were met in the initial delivered system. The system has captured the required technical expertise and also provided a significant improvement in productivity. The cabling capabilities of EXCABL are such that only a small percentage of actual cabling design tasks cannot be handled. Since the end product is a cabling installation drawing, any EXCABL solution can be manually modified or augmented to produce a more acceptable solution. The productivity improvement realized

by this new capability is such that a typical mission cabling manifest, formerly taking a few labor-intensive days for several cabling engineers, now takes only a few minutes.

The expert system portion of the operational version of EXCABL was implemented using Production Systems Technology's C-based version of OPS83. The remaining portion was implemented using C. It is currently hosted on a CAD/CAM interfacing DEC MicroVax II system and integrated into the operational environment. The literature contains documentation of an early prototype version of EXCABL (Reference 1), problems associated with converting from a development system to a delivery system (Reference 2), and a case study of the development effort and lessons learned (Reference 3).

PAYLOAD BAY CABLE INSTALLATION TECHNICAL ORDERS

The Problem

The cabling layout solution schematic produced by EXCABL is only one of many Space Shuttle planning and design products necessary to accomplish the actual electrical services provisioning of its payload bay. Among the other products required are the installation configuration TO's for the cables and related hardware devices. These TO's contain the detailed instructions that are used by the payload integration crew to perform the actual cable and hardware device installation. The cabling TO's required for each flight are unique and dependent on the cabling solution generated by EXCABL.

In order to increase productivity, the concept of modularization was developed by the cabling design engineering group. This concept is to reuse previously generated TO's whenever possible, thereby, eliminating the need to repeat labor-intensive documentation for the same installation. Implementation is accomplished by assigning basic TO numbers for each device, connector or cable installation and by assigning dash numbers for the different configurations. If an existing TO is not available, a new TO is generated, and a new dash number is assigned. The modularization, or reuse concept, is only made feasible by the standardization of cables, connectors, devices, mounting positions, etc.

Identifying the set of all reusable cabling installation TO's for each given mission is a recurring integration problem. Since the set of cabling TO's for each mission is dependent on the EXCABL solution, any automation of this process should interface easily with existing EXCABL outputs. Furthermore, maximum usage should be made of any intermediate information generated by EXCABL to support its final products. EXCABL changes to support the TO identification and generation process would be undesirable and should be kept to an absolute minimum. The desirability of integrating this process into the existing work environment, while cooperating with the EXCABL process, placed serious constraints on the design and development of any automated system solution.

Simply stated, the problem was to develop an automated system that has the capability to identify and generate a list of all TO's required to perform the payload cabling installation for any Space Shuttle mission. If any required TO does not currently exist, that fact should be identified by the system to the user in order for the deficiency to be corrected and the process completed.

The Solution

There were two basic motivators for this project: the demonstrated success of the EXCABL project and the practicality of automation based on the new concept of modularization and reuse. Furthermore, it was assumed that application of the experience and techniques gained from the EXCABL project would make this a low risk development effort (Reference 3.) Those assumptions proved to be correct in practice, and the entire development effort was

straightforward and completed without any major problems within six months of project initiation.

An expert system, called the Expert Drawing Matching System (EXMATCH), was placed in operational usage in January of 1988. The EXMATCH system has successfully automated the payload bay cabling installation TO generation task. Closely integrated with the EXCABL system, the cabling solution provided by EXCABL is automatically input to EXMATCH and a master listing of all required payload cabling installation TO's is generated. If a required TO does not currently exist, the system not only identifies this deficiency, but also suggests a similar drawing (best match) that may be modified to satisfy the deficiency. An example of a typical output is shown in Figure 4.

To facilitate this system, an initial data base containing all current payload cabling installation TO numbers was constructed. For user convenience, the interface to maintain this data base was made an integral part of the EXCABL system. The development of the initial TO documentation and maintenance of the TO data base are the only functions not fully automated. Modifications to EXCABL were limited to providing the user with an EXMATCH option selection capability, providing and relinquishing program control to EXMATCH, and providing the user interface for maintenance of the TO data base. An overview of the integration of EXMATCH and EXCABL is shown in Figure 5.

Installation T.O. Dash Number Listing

Device Installation Dash Numbers			
panel kind	panel side	panel fwdx0	dash number
sip	left	1043.560	*** no dash number *
pat	right	1141.770	M072-710427-058
egress	left	1153.770	M072-710425-077
foldcover	left	631.750	M072-710426-001
longeron	left	1140.000	M072-710005-019
standoff	left	1140.000	M072-710006-019
instl payload, dynamics, (m0771a)			M072-754117
wire tray, cover and gap filler			M072-710012-002
		...	
		...	
Connector Installation Dash Numbers			
sip	left	1043.560	
number of connectors: 1			
j402			
number of required unmounted connectors: 1			
j401			
dash number: *** M072-710028-024 ***			
Cable Installation Dash Numbers			
number of devices with cables: 1			
sip	left	1043.560	
number of cables: 2			
j402	p402	46p77w421	
j401	j401	46p74w7527	
number of devices with no cables: 1			
egress	left	1041.990	
*** no dash number ***			
		...	
		...	
Best Matched Dash Numbers			
sip	left	1043.560	
is very close to the following installation:			
sip	left	1045.000	M072-710305-001
		...	
		...	

Figure 4. Typical EXMATCH Listing

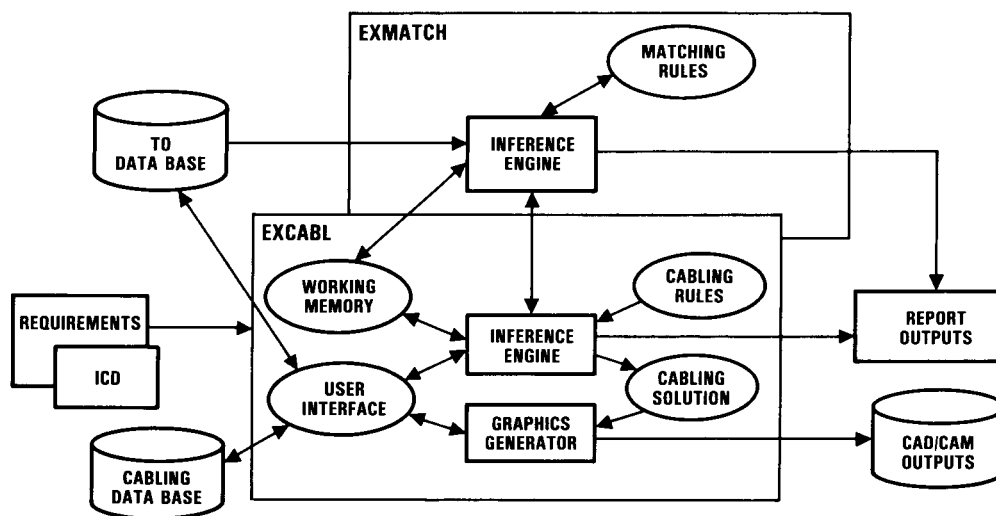


Figure 5. EXMATCH/EXCABL System Integration

The expert system portion of EXMATCH was implemented using Production Systems Technology's C-based version of OPS83. The remaining portion was implemented in DOD's registered trademarked language, Ada. EXMATCH is currently cohosted with EXCABL on CAD/CAM interfacing DEC MicroVax II systems in the cabling design work place.

TECHNICAL ORDERS FOR TOTAL PAYLOAD INTEGRATION

The Problem

The Space Shuttle payload integration planning and design process culminates in the provision of a complete set of TO's containing all of the installation instructions needed to accomplish the total payload bay accommodation and installation task. To assist in the planning and installation process, a complete list of all applicable TO's for a mission is specified in a single document, the Mission Equipment Cargo Support Launch Site Installation (MECSLSI) drawing. Since each Space Shuttle mission is basically unique and many design changes occur subsequent to initial payload manifesting, the identification of all required TO's for the production of this drawing constitutes a continuing and complex integration problem.

Mission requirements are categorized as either mission-common or mission-unique. Mission-common requirements are those requirements that, once established, are standardized for all future missions. Mission-unique requirements are dependent on each mission's objectives. Since the payload manifest is basically unique, the payload cabling layout schematic TO produced by EXCABL is mission-unique. However, it has been estimated that 90 percent of mission requirements fall in the mission-common category.

On the basis of flight requirements documentation, interface control documents (ICD's), mission-unique TO's, common TO's, similarities to previous missions, etc., the mission-MECSLSI development expert utilizes heuristic knowledge to generate the required drawing. Due to the

preponderance of mission common TO's, if a design automation system could be developed to produce an initial MECSLSI containing only those TO's, labor requirements would be reduced considerably.

The Solution

A feasibility study was initiated in Fiscal Year 1987 to determine the practicality of developing an expert system to automate the production of the initial MECSLSI drawing. As a consequence of positive study results, it is expected that development of an expert system based design automation tool, the Technical Order Listing Expert System (EXTOL), will be started in the near future. Not only will EXTOL produce the initial MECSLSI drawing; but using heuristics and data from previous missions will assist the user by producing a list of the best matches for missing TO's in the mission-unique category. If a close match cannot be found, it will identify that fact and provide further assistance to the user by presenting essential configuration information. An example of the information processing of EXTOL is shown in Figure 6. It is reasonable to expect that a working prototype could be produced in Fiscal Year 1988, and an operational system delivered in Calendar Year 1989.

TOTAL PAYLOAD AND CARGO INTEGRATION AUTOMATION

The EXCABL system produces the mission-unique cabling layout schematic TO product. EXMATCH uses information generated by EXCABL in its solution process to augment its knowledge and produce a list of all existing TO that will be required to accomplish the cabling installation. If a required TO does not already exist, a best match or similar existing TO is identified. When all of the required TO's are generated and this information is input to EXMATCH, a complete list of cabling TO's is generated. Together, these data will be furnished as electronic inputs to the EXTOL system currently under consideration. These improvements should allow EXTOL to produce initial MECSLSI drawings that are over 90 percent complete. For TO's identified as needed, but not in existence, best match and configuration information to greatly facilitate their generation will also be produced.

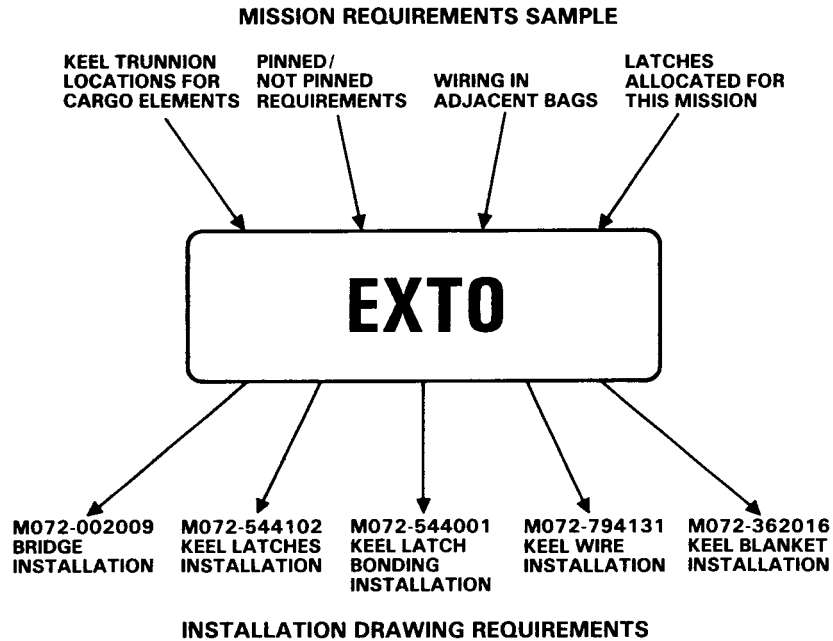


Figure 6. EXTO Processing Example

Together, EXCABL, EXMATCH, and EXTOL constitute a real-world demonstration of the feasibility and benefits of applying expert systems technologies to the payload bay integration automation problem. Two of these systems have been integrated into the engineering work environment and cooperate to automate the overall payload integration management task. The third, when completed, will be integrated with the other two to further the goal of total payload bay integration automation. Since each expert system feeds its outputs directly to its successor, the productivity improvements of the group as a whole are greater than individual stand-alone systems could achieve. An overview of the integration management of these expert systems is shown in Figure 7. Using these systems as the core, other expert systems aimed at supporting the automation goal are in the concept development stage.

SUMMARY

The high level of flexibility to handle diverse payloads provided by NASA's Space Shuttle orbiter presents recurring payload and cargo integration problems. Expert system technologies are being applied to these problems to provide a level of automation that was previously unrealizable.

EXCABL is an example of a highly successful, nontrivial, demonstration of the applicability of using artificial intelligence techniques to solve real-life design automation problems. It met the design automation objectives of reducing engineering-labor hours and end-to-end process time, captured corporate technical planning and design expertise, and demonstrated that expert

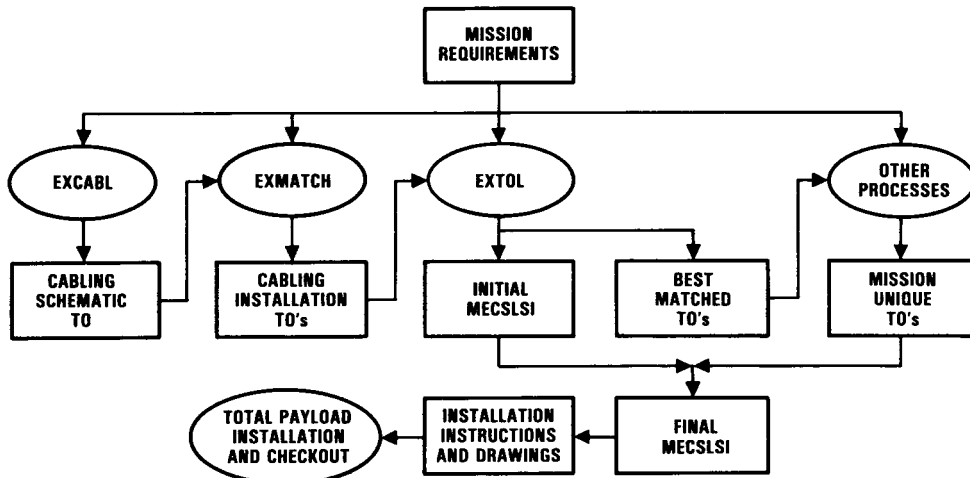


Figure 7. Payload and Cargo Integration Management

systems methods can successfully be integrated into existing operational systems.

EXMATCH is a second example of the appropriate application of expert system technologies to solve automation problems. Its inception was driven by the success of the EXCABL system. It was implemented quickly without major development problems and successfully integrated into the existing work place along with EXCABL.

The development of another expert system based design automation tool (EXTOL) is currently under consideration. These three separate systems when implemented will work together to support overall payload integration management automation.

ACKNOWLEDGEMENTS

The author would like to thank Gil Nixon, Beshad Rejai and Chuck Giffin for their contributions and support.

REFERENCES

1. Saxon, C.R., and Schultz, R. "EXCABL: An Expert System for Space Shuttle Cabling," AI-85, *Proceedings of the First Artificial Intelligence and Advanced Computer Technology Conference*, Long Beach, California, 1985, pp. 127-140
2. Saxon, C.R. "Converting a Demonstration to a Delivery Expert System: Lessons From EXCABL." *Proceedings of IEEE Westex-86 Expert Systems Conference*, Anaheim, California, 1986
3. Morris, K.E., Nixon, G.A. and Rejai, B. "EXCABL—Orbiter Payload Bay Cabling Expert System, A Case Study," *Proceedings of IEEE Westex-87 Expert Systems Conference*, Anaheim, California, 1987

ROBOTIC TELEPRESENCE APPLICATIONS FOR THE AIR FORCE

Terrell Scoggins
Wright-Patterson AFB

(Paper not provided by publication date.)

PRECEDING PAGE BLANK NOT FILMED

IDEAL: A METHODOLOGY FOR DEVELOPING INFORMATION SYSTEMS

Ken H. Evers
SofTech, Inc.
3100 Presidential Dr.
Fairborn, Ohio 45324-2039

Robert F. Bachert
Armstrong Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

As a result of improved capabilities obtained through current computer technologies, application programs and expert systems, Enterprises are being designed or upgraded to be highly integrated and automated information systems. To design or modify Enterprises, it is necessary to first define what functions are to be performed within the Enterprise, identify which functions are potential candidates for automation, and what automated or expert systems are available, or must be developed, to accomplish the selected function. Second, it is necessary to define and analyze the informational requirements for each function along with the informational relationships among the functions so that a database structure can be established to support the Enterprise.

To perform this type of system design, an integrated set of analysis tools are required to support the information analysis process. The IDEAL (Integrated Design and Engineering Analysis Languages) methodology provides this integrated set of tools and is discussed in this paper.

INTRODUCTION

The information age is upon us. More than half of the people going to jobs every workday are primarily information workers and the percentage is growing as the industrial sector becomes less labor intensive. Information in this paper is defined as being the integration of data and information. Those who are responsible for supporting these information workers with

automated tools have steadily raised the importance of information. It has been transitioned from a much overlooked role in the early days of file processing to the lofty position of a resource deserving of the same attention as that given to the management of people, money, and materials.

The approach which recognizes that information is at the heart of our information systems design effort is identified as Information Resource Management (IRM). Before developing a particular view or implementation of an IRM system, an understanding must be gained as to what the information is, how it exists, and how it is used within Enterprises. From this understanding, the proper use of automated tools, computers, and robotics, can be perceived and implemented. Without this understanding, management decisions related to information handling capability will simply be directed at hardware and software, and will have gained little advantage over previous methods.

Each person working in an Enterprise perceives a subset of the Enterprise's information base within the context of the tasks he perform. This perception is called a "user view." (It is the total integration of all the user views that contribute to making up the information base of the Enterprise.) In order for a system to serve all users effectively, the information must capture the inherent nature of data as it exists and not pay particular attention to any one user view. Therefore, the abstraction of data reality captured by the model must be representative of the inherent structure of the data as it exists from an overall perspective and not be biased by any one perspective.

To support an effective IRM effort requires an integrated set of tools and techniques necessary to build and analyze an information model. This paper discusses a methodology called IDEAL which is composed of such a set of tools and techniques.

DISCUSSION OF IDEAL

The IDEAL methodology has been formulated and applied over a five year period. It has proven to be an effective approach through the integration of proven techniques (Figure 1). IDEAL provides a top-down, structured technique to define and document the system of interest; a knowledge engineering technique to collect and organize system descriptive information; a rapid prototyping technique to perform preliminary system performance and effectiveness analysis; a sophisticated simulation language to perform in-depth system performance analysis; and a data modeling technique to perform information analysis for developing data base design.

IDEAL is composed of three techniques that have been developed and proven as stand-alone capabilities. Among these are IDEF(0), IDEF(1), SADT, and SAINT. The IDEF(0) technique was developed to provide a structured approach to defining or bounding the system of interest. IDEF(0) provides this definition initially as a general view in terms of the activity the total system is to accomplish, what is produced by the system, what materials and information are used by the system, and what criteria control how the

system performs its activity. Through a knowledge engineering and a graphical presentation technique, IDEF(0) provides the structure for gradually detailing out the general system activity into subactivities, the informational requirements for each activity, and the informational relationships among the activities.

The SAINT (System Analysis for Integrated Networks of Tasks) simulation language was developed to represent the dynamics performance of a system by defining and linking the activities performed within the system and by representing the performance relationships existing among the activities.

The IDEF(0) and SAINT techniques are closely related in that the activities defined by IDEF(0) are the same as those represented in SAINT. Also, the data relationships among the activities in both techniques are the same. The primary difference between the two techniques is that IDEF(0) is a static representation of the system whereas SAINT provides a dynamic representation of the system. To form the link between the static and dynamic representations, the Performance Data Base (PDB) was formulated. The PDB is filled by specifying dynamic information for each subactivity represented in the IDEF(0) model so as to define the performance characteristics of each activity along with the performance relationships among the activities. The dynamic representation of the system is then developed through the SAINT language by forming the basic network through the IDEF(0) model onto which the dynamic information from the PDB is integrated.

IDEF(1) is the third tool within the IDEAL methodology. IDEF(1) is a proven data modeling technique. The goal of IDEF(1) is to identify information that exist within the Enterprise, how and where the various aspects of the information exists, and the grouping of the information through a series of entity / attribute relationships. The resulting IDEF(1) model provides a means for understanding how the information should be organized and moved through the Enterprise and forms the basis for designing the databases that will eventually be developed within the Enterprise.

The one feature which provides IDEAL with its power as an analysis tool is its utilization of a team effort. IDEAL accomplishes this integration through its knowledge engineering

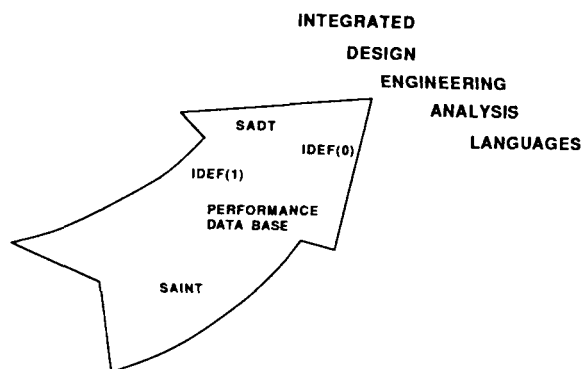


Figure 1. IDEAL Overview

capability. The process of developing the functional model and the system performance characteristics is accomplished through an interviewing procedure during which the analyst collects information from system experts. The analyst then integrates the information and represents it in terms of the functional model and the system performance description documented in the PDB. This integrated process helps to identify discrepancies among information sources as well as missing information about the system. This integrated information is then reviewed by the system experts in order to verify the analyst's understanding of the system, to add additional information, or to clarify conflicts among the experts.

DEVELOPING AN ENTERPRISE

Performing an informational analysis on an Enterprise for the purpose of incorporating automation and expert systems is a seven-step procedure. There are slight variations in the procedure when applied to the initial design of an Enterprise as opposed to being applied to the upgrade of an existing Enterprise being upgraded to incorporate new technologies. When designing new Enterprises, the definition of the activities to be performed, the information definitions, and the performance specifications must be established by the designers through an understanding of what is needed to be done and what technology could be utilized to form the "TO-BE" model. When upgrading an existing Enterprise, the requirements of the upgraded Enterprise are defined by the upper-level management and used to form the "TO-BE" model. Assuming the current Enterprise will be modified as opposed to being replaced, the characteristics of the current Enterprise are defined and understood through the development of "AS-IS" models. By comparing the "TO-BE" and the "AS-IS" models, a plan of attack is then defined which will be the roadmap to upgrade the current Enterprise into the desired Enterprise.

An Enterprise is a system composed of facilities, personnel and equipment, within which products are produced through a set of processes made up of activities. A significant portion of activities within an Enterprise are those that process material goods. The material used within each activity is controlled by identification codes

identifying the materials, where the materials are located, the activities to be performed on the material, and the materials' destination following the completion of each activity. Within this definition, an Enterprise can be any one of a wide variety of facilities. For example, a manufacturing facility transforms raw materials into subcomponents and the subcomponents are assembled into final products. As the raw materials and subcomponents proceed through the manufacturing facility, they are identified and tracked through identification numbers.

In a similar manner, an Enterprise can be an office environment. Within the office, a requested report involves the production, selection, and integration of information so as to tell a desired story. The general data, or raw material, is available to the authors through a defined access procedure and is manipulated to form segments of the report which, in turn, are integrated to form the final report, or product.

Performing the information analysis (requirements analysis) required to develop an efficient, automated Enterprise is accomplished through the following steps:

1. Interview Managers / Designers,
2. Identify Enterprise Functions,
3. Collect Performance Information,
4. Simulate the Enterprise Operation,
5. Identify Informational Relationships,
6. Perform Technology Assessment, and
7. Develop Physical Model.

The relationship among these tasks are illustrated in Figure 2.

Step 1. Interview Managers / Designers

Using IDEAL's knowledge engineering aspects of IDEF(0) and IDEF(1), the interview of the top-level management is the most critical phase because it forms the requirements analysis for the Enterprise development in terms of the limits, direction, scope, and authority. Through these interviews is determined the current mission goals for an existing or new Enterprise and future mission goals of the Enterprise, an identification of the critical factors for success within the Enterprise, and a top-level understanding of the activities performed within the Enterprise. From Step 1

ORIGINAL PAGE IS
OF POOR QUALITY

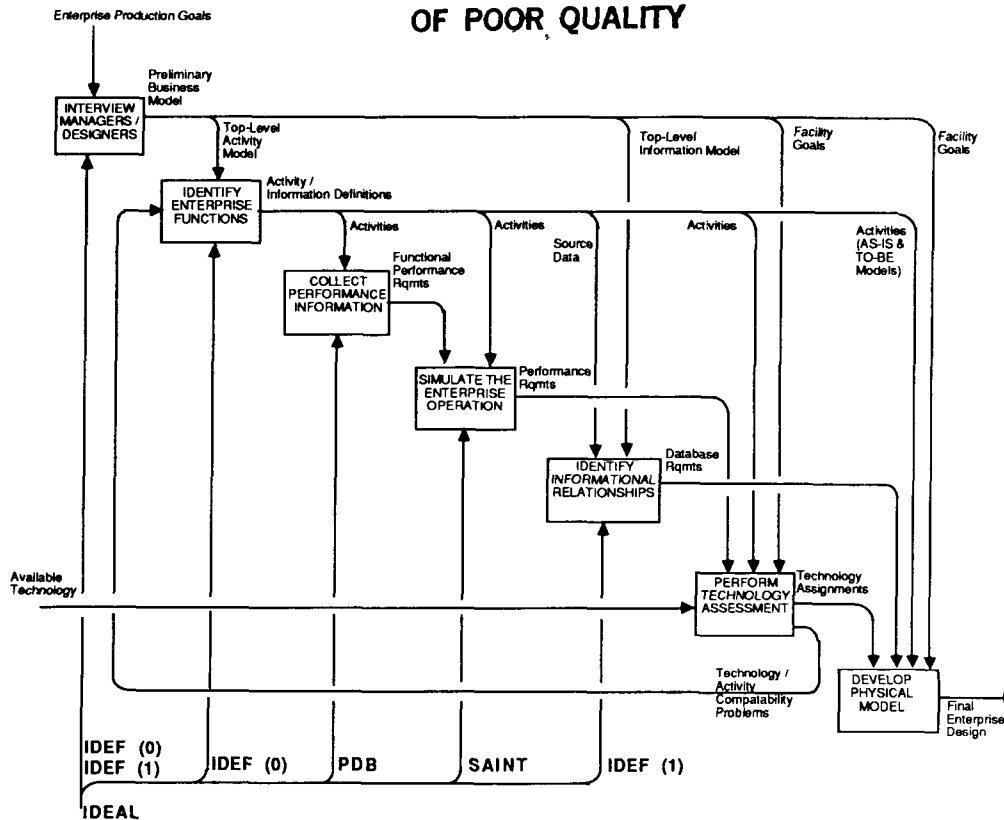


Figure 2. The Approach for Developing an Information System

is produced a top-level model (referred to by some as a Preliminary Business Model) that does not include specifications for technology and organization, and the viewpoint and purpose for performing the activities for the overall Enterprise development effort.

Step 2. Identify Enterprise Activities

Through the continuing use of IDEAL's knowledge engineering capability, the top-level activity model developed during Step 1 is expanded through a series of interviews with individuals, at various levels or Enterprise responsibility to gain an understanding as to what activities must be performed to produce the desired products and what informational relationships exist among the activities. This expanded activity model is developed to a level of detail sufficient to respond to the purpose identified in Step 1. This activity model is often referred to as a logical model because, like the Preliminary Business Model, it does

not include specifications for technology, individuals, equipment, and organization. It provides a description of what the Enterprise does and what information is used rather than how the activities are accomplished.

Step 3. Collect Performance Information

The goal of this aspect of the IDEAL methodology is to begin specifying performance requirements for the Enterprise. Using the activity model as the activity definition, Step 3 overlays performance information onto each identified activity. The performance information identifies such items as performance times desired for each activity and backlogs allowable at each activity. The dynamic information defined in this step represents an initial set of performance requirements for subsystems that will eventually perform each activity.

Step 4. Simulate The Enterprise Operation

Within the IDEAL methodology, the capability has been developed which allows for the development of a simulation model by integrating the activity model and the performance information through the use of a network simulation language. By exercising the simulation model, insight into the performance characteristics of the Enterprise can be obtained. This insight includes an analysis of the workload at each activity, chokepoints within the Enterprise processing based on the process performance requirements specified in Step 3 as a function of potential automated capabilities. By exercising the simulation model, a more detailed set of performance requirements can be specified for each activity and potential areas for automation can be identified.

Step 5. Identify Informational Relationships

Having formed a solid understanding of what the Enterprise is to accomplish through the previous four steps, Step 5 changes the direction of the analysis goals by concentrating on the information aspect of the Enterprise. The goal of this step is to describe the Enterprise from an information viewpoint and the data required by the Enterprise, but like Steps 2, 3, and 4, does not depend on either the organizational structure or technology.

The Information model is analyzed to identify natural groupings of information. The groupings will eventually become the basis for the design of the databases to be developed in support of the overall Enterprise. Subsequent analysis results in the database design for each application system along with how and where the information will reside in the specific hardware and software environment.

Step 6. Perform Technology Assessment

The objective of Step 6 is to analyze the technologies, software, hardware, automation, expert systems, etc., available in an attempt to satisfy the Enterprise performance requirements established during the previous steps. As mentioned earlier, Step 6 is not performed using the tools within the IDEAL methodology but rather uses the models developed by the previous sets as the basis for establishing the requirements for determining

what types of automation, technologies, and expert system should be considered and a preliminary judgement as to which of these will satisfy the performance requirements of the Enterprise. The results of the Step 6 assessment may be fed back into the activity model if it is found that no technology exists to accomplish a given activity. This will require a decision to either develop the capability or revise the activity so that the necessary technology does exist. It may also be realized that a given technology is available to accomplish a group of activities. Therefore, the activity model may be modified to combine the activities to better represent how the automation will be implemented into the Enterprise.

Step 7. Develop the Physical Model

The objective of this step is to transform the results of the analysis performed up to this point into a physical representation or design for the Enterprise. This involves the designation of how each activity will be performed, how the databases will be structured, and how the organizational structure for the Enterprise will appear.

The procedure for accomplishing this will vary depending on whether a new Enterprise is being developed or whether an existing Enterprise is being upgraded. When designing a new Enterprise, only the considerations for what the Enterprise should be need to be considered with respect to the activities performed, the technologies available, the database requirements, and the goals of the Enterprise.

When upgrading an existing Enterprise, it would be impractical to simply replace all of the existing information processing procedures and capabilities. Instead, automation and information systems must be integrated at appropriate points in the Enterprise. To accomplish this, an analysis must be made between the current and desired Enterprise design. By understanding these differences in terms of activities performed, database requirements, and available techniques, a plan is developed which specifies the approach to be used in upgrading the Enterprise. Through the information generated with the IDEAL methodology, such factors as the criticality of a given activity, impact of automating one activity on those activities interfacing with

it, and the overall modernization goals of the Enterprise can be considered.

CONCLUSIONS

IDEAL is a proven, general purpose methodology for modeling a wide variety of system types. IDEAL forms the foundation needed to understand, document, and analyze the requirements for the Enterprise. IDEAL's power lies in its integrated set of techniques which utilize a knowledge engineering approach in conjunction with a top-down modeling and simulation approach to collect, integrate, and verify the information within an Enterprise through a team effort. Based on the understanding of the enterprise provided by through the IDEAL technique, areas for automation and improved information processing can be identified and eventually implemented as appropriate to satisfy the modification goals of the Enterprise.

REFERENCES

- SofTech, Inc., "SAINT Performance Assessment Model of a SAM System (SPAMSS); Analyst Manual", August 1984.
- Cooper, W.M., "Using SAINT in Performance Analysis of Complex Hardware/Software Systems", Proceedings of the IEEE 1985 National Aerospace and Electronics Conference (NAECON), Dayton, Ohio.
- SofTech, Inc., "High Speed Ring Bus (HSRB) Protocol Analysis", June 1987.
- Evers, K.H., Bachert, R.F., "SADT: An Effective Tool For Knowledge Acquisition", Human Factors in Organizational Design and Management - II, August, 1986, Vancouver, B.C., Canada.
- Bachert, R.F., Evers, K.H., Hoyland, C.M., & Rolek, E.P., "IDEF/SAINT SAM Simulation: Hardware/Human submodels", Proceedings of the IEEE 1983 National Aerospace and Electronics Conference (NAECON), Dayton, Ohio.
- Bachert, R.F., Evers, K.H., Santucci, P.R., "SADT/SAINT: Large Scale Analysis Simulation Methodology", Proceedings of the 1981 Winter Simulation Conference.

MODEL-BASED REASONING FOR SATELLITE AUTONOMY

Lorraine Fesq.
TRW Systems

(Paper not provided by publication date.)

DEVELOPMENT OF A PERSONAL-COMPUTER-BASED INTELLIGENT TUTORING SYSTEM

Stephen J. Mueller
Computer Sciences Corporation
16511 Space Center Blvd.
Houston, Texas 77058

A large number of Intelligent Tutoring Systems (ITSs) have been built since they were first proposed in the early 1970's. Research conducted on the use of the best of these systems has demonstrated their effectiveness in tutoring in selected domains. Computer Sciences Corporation, Applied Technology Division, Houston Operations has been tasked by the Artificial Intelligence Section at NASA/Johnson Space Center (NASA/JSC) to develop a prototype ITS for tutoring students in the use of the CLIPS [1] language: CLIPSIT (CLIPS Intelligent Tutor). For an ITS to be widely accepted, not only must it be effective, flexible, and very responsive, it must also be capable of functioning on readily available computers.

While most ITSs have been developed on powerful workstations, CLIPSIT is designed for use on the IBM PC/XT/AT personal computer family (and their clones). There are many issues to consider when developing an ITS on a personal computer such as the teaching strategy, user interface, knowledge representation, and program design methodology. Based on experiences in developing CLIPSIT, this paper reports results on how to address some of these issues and suggests approaches for maintaining a powerful learning environment while delivering robust performance within the speed and memory constraints of the personal computer.

CHOOSING A DEVELOPMENT ENVIRONMENT

One of the major goals of Intelligent Computer Aided Instruction (ICAI) is to improve the quality of education and training especially for the average and below average student. For this to happen, educationally effective tutors must be made widely

available to end users, be affordable, and run on inexpensive computers. Strategies for cost effective development of ICAI programs for small computers include choosing a language and an environment for developing the ICAI based on availability, language and development costs, and performance requirements.

The delivery environment chosen for CLIPSIT was the IBM PC/XT/AT family of computers using the languages CLIPS, C, and a commercially available graphics package. There were a number of reasons for which CLIPS was chosen. The first reason is that CLIPS is an expert system building tool written in C which can be compiled to create a runtime executable. A typical development strategy for ICAI is to develop a prototype first using an artificial intelligence (AI) language or programming environment and then rewrite the runtime programs in a general purpose language (such as C) to achieve optimal performance and transportability and to minimize distribution costs [2]. By using the CLIPS language, this rewriting phase could be eliminated thereby saving valuable resources.

A second reason CLIPS was chosen as the development language is its availability to developers and students. CLIPS is available at no cost to anyone currently working on a federal government contract. Since the intention is to distribute a copy of CLIPSIT along with the CLIPS language, the student would have easy access to the required software.

A third reason is the high performance and extensibility of CLIPS. External functions can easily be written in C and called from the CLIPS production rules. The ability to write user defined functions was invaluable to the teaching expert whose job it was to

manipulate the windows and menus of the user interface as well as communicate with the tutorial parser.

The IBM PC family of computers was also chosen as the development and delivery vehicle for a number of reasons. Most students have access to personal computers either at home or school or business. Availability is one of the most important requirements for realizing the utility of an ITS. Inexpensive personal computers now have the power required for many ICAI applications whereas previously most applications were developed on specialized AI workstations. Intelligent tutors are just one example of ICAI where remarkable progress is being made on personal computers. As processing power and memory capacity costs continue to decline, inexpensive machines will be capable of supporting significant ITSs. Finally, the CLIPS language is a very effective tool on the personal computer and the C language compilers necessary to take advantage of its extensibility are readily available.

CLIPSIT APPROACH

CLIPSIT is a knowledge-based system which tutors students in the concepts and syntax of the CLIPS language. The primary goal of the CLIPSIT tutor is to provide a proof of concept which can demonstrate that a usable tutor can be developed on a personal computer. Once completed, CLIPSIT will provide approximately 10 lessons with each lesson containing about 10 problems. The program currently consists of about 75 generalized rules written in CLIPS and a high quality user interface written in C and a commercial graphics package. Rules comprising CLIPSIT represent the following typical functional modules in an intelligent tutoring system: a domain expert, a diagnostic expert, a teaching expert, and a student model. The functional components of CLIPSIT are shown in (Figure 1) and described in more detail on the following pages.

The student interaction with the tutor occurs through a two-window user interface which presents a problem description in one window and accepts student responses in the other. Student responses are analyzed by the tutor and compared against expected student responses. The student is notified immediately of any discrepancies found in his syntax or logic by the tutor. Student responses which deviate from the expected response are compared to anticipated errors in a bug

catalogue. Those bugs which are not present in the bug catalogue are handled by special diagnostic rules. Each problem presented to the student is stored in a separate file along with the appropriate lesson text and is presented upon request by the student model. These files contain such information as expected responses, expected errors and error messages, teaching strategies, and problem description facts which describe the allowable variations to the "correct" expected responses the student is permitted to make. At the completion of a problem, all facts pertaining to the current problem are purged and a new problem is presented. By modularizing problems, a general set of (domain) rules can exist to analyze each specific set of problem facts. Modularization has the additional benefit of eliminating the need for problem specific "buggy rules" [3] which would greatly increase the size of the rule base and inhibit performance.

DOMAIN EXPERT

The domain expert contains the knowledge that the student needs to learn. This knowledge is used to solve problems generated by the teaching expert in order to provide a basis for error analysis by the diagnostic expert. As tokens are entered by the student, (tokens are definable characters that serve to delimit the contents of an input string) they are checked against the predefined expected response for that token. More specifically, the domain expert reads in problem description facts as well as a skeleton solution containing the expected student responses. As long as the student response continues to match the expected response, the student is left alone. Should the student response differ from the expected response, additional analysis is made to determine if the response is indeed an error or simply another equally valid approach to solve the problem. When the student response is determined to be valid, the fact database is updated to reflect the current approach. However, if the student response is determined to be an error, then an error fact is created for the diagnostic expert to analyze.

Many features are built into CLIPSIT which provide the student with the freedom to use his own creative instincts to solve a problem. This is a tremendous advantage over forcing the student to follow a strict one-to-one mapping between a student response and an expected system response. One example of the flexibility of the tutor is in regard to the naming of pattern variables in CLIPS. Should the student choose to name a variable something other than what the system expects, this should be (and is) allowed. Also,

in many cases, the order of patterns in a rule is arbitrary. By reading the problem description facts from the problem file, the domain expert knows which patterns can be interchanged. Additionally, there are times when the tokens in a pattern can be arranged arbitrarily such as when the student is applying logical field constraints. All of these variations to the expected system responses are supported given that the student's solution is an equally valid solution to the problem.

Since a fast response time is of utmost importance for a tutor to hold the student's attention, only one skeleton solution is provided. That is, there is only one acceptable logic path for the student to follow. In order to make one skeleton solution span the entire set of possible solutions, the problem must be very narrow in scope. All of the problems in CLIPSIT are worded in such a way that there is basically only one way to solve the problem given the normal creative variations mentioned above. By limiting the scope, the number of solutions becomes very manageable and the expected student responses can be predicted with great accuracy. The obvious benefit of this approach is the improved response time obtained by requiring less rules and facts to describe and analyze the problem.

DIAGNOSTIC EXPERT

The diagnostic expert provides error analysis for the student responses. Once an error has been encountered by the domain expert, that error is intercepted by the diagnostic expert to try and determine the cause of the student's misconception. Based on the error analysis, the diagnostic expert hypothesizes what misconceptions the student may have. These hypotheses are recorded in the current state of the student model.

There are three sources of error diagnosis in CLIPSIT: a bug catalogue containing a list of expected errors for a particular problem, a set of logic checking rules which uses the problem description information to analyze the consistency of the student response, and a catch-all set of rules which detects simple syntax errors and misspellings.

The bug catalogue is generated from experience gained in teaching classes on CLIPS and noting various student errors and misconceptions. The advantage of this approach is that, although cumbersome, empirical data such as this can be easily obtained from these

classes.[4] Approximately fifteen CLIPS classes have been presented to date to draw upon for information. The bug catalogue consists of a table of expected errors for each token in the student response. For every expected error in the bug catalogue, there exists a corresponding diagnostic message. In this way, a very specific diagnostic message is presented to the student directly applicable to the current problem.

The logic checking error rules are basically inverses of many of the domain rules. These rules analyze the expert skeleton solution and problem description facts and can recognize inconsistencies in the student thought process. For example, if the student had used a variable to represent the price of a pair of shoes in one pattern, and then tried to use that variable to later represent a coat, the error would be caught. Every diagnostic rule has the ability to generate specific error messages for the particular student response being analyzed. The general logic checking rules contain a skeleton error message with appropriate slots to be filled in with diagnostics specific to a particular problem.

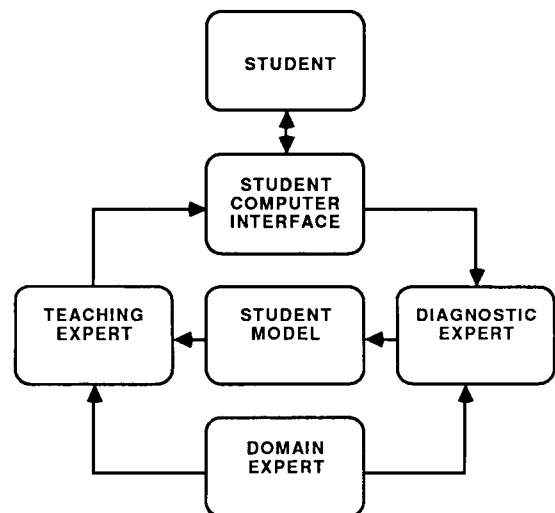


Figure 1. TYPICAL INTELLIGENT TUTORING SYSTEM (ITS) ARCHITECTURE

The third set of diagnostic rules provides a low level of analysis capable of detecting simple syntax errors and misspellings. Their main purpose is to provide a catch-all for errors which have filtered through the tutor and remain undiagnosed. The objective of the knowledge engineer is to minimize the number of times these rules are required to execute by improving the effectiveness of the bug catalogue and logic checking rules.

The error messages provided by the three diagnostic rule sets vary in their levels of specificity. The expected error rule set provides the most specific messages followed by the logic checking rules and finally the catch-all rules. Since the goal of the diagnostic system is to provide the most meaningful diagnosis available for a student response, a hierarchy of priorities among the three diagnostic rule sets was created. Whenever an error is detected, there is the possibility that a rule in all three diagnostic rule sets could be activated. Since the nature of the error messages becomes more general as the priority of the diagnostic rule set decreases, it is important to enable the rule with the best (most specific) message. By implementing this priority system the student is always assured of getting the best diagnosis possible. One advantage of this three level error detection system is that the knowledge engineer can continue to collect expected errors for the bug catalogue as the tutor matures and easily add them to the system at any time. Improving the bug catalogue's ability to detect errors allows the tutor to divert the error analysis from less specific logic checking and catch-all rules to the most specific expected error rules.

TEACHING EXPERT

The teaching expert is a set of specifications of what instructional material the tutor should present and how the material should be presented. All problems in CLIPSIT have a tutoring strategy specified within the problem description facts. Although this approach restricts adapting the problem presentation to more closely match the ability of a particular student, the immediate benefit of less code and faster response time achieved by not having to compute a presentation strategy, is a viable tradeoff on personal computers. There are three basic strategies available to the knowledge engineer in CLIPSIT for presenting a problem. Selection of one of these three strategies depends on the complexity of the problem to be presented and the current lesson.

The first strategy is to provide the student with examples and a list of candidate responses available for use in the solution. Since the first hurdle of the student is determining what the immediate goal is, that is, determining the desired response from the instruction [5], a list of solution components provides the student with adequate goal reinforcement. Additionally, many teachers agree that certain classes of students are competent to solve sets of problems only after an example is done for them. By reinforcing the goal

with examples, performance improves significantly. From the standpoint of the diagnostic expert, this approach is also beneficial since it now has a clear understanding of what the correct solution must be. For example, if the student is given a problem description and asked to generate a fact to represent some aspect of the problem, the student could generate many valid representations by choosing endless different names for the same item. By naming the tokens in the problem for the student, a finite and manageable solution set is now available and the student has gained valuable insight by example.

A second strategy for presenting the problem is to use a template. Problems can be presented to the student as a puzzle where the tutor supplies some of the solution pieces and the student supplies the rest. The earlier lessons and problems would provide more pieces of the puzzle than the later problems. The template is very useful, once again, for reinforcing the students understanding of what the problem is really asking for. It also provides a means for the teaching expert to lead the student down the desired solution path, thereby providing for faster and easier error diagnosis.

The third strategy, if it can be called that, is to simply turn the student loose to take whatever approach he or she chooses. Since this method is by far the most difficult to provide diagnostic analysis for, the problem must be worded in such a way that the correct solution can be readily anticipated by the tutor.

All strategies have the benefit of coaching messages generated by the diagnostic rules. A student is given three attempts to provide the anticipated response for the solution. If the student has not determined the correct response by the third attempt, he is given the answer.

STUDENT MODEL

The student model is the representation of the student's understanding of the domain knowledge as perceived by CLIPSIT. The student model is used to assess the student's comprehension of the problem goals and to make decisions about what strategy should be followed to correct any perceived student misconceptions. By comparing the student performance against the expected responses of the tutor, (also known as the "overlay model" [6]), CLIPSIT can determine which teaching goals the student has failed to grasp.

Every problem file in the problem set available to the student contains information about the teaching goals it is trying to present. Certain problems are designated by the problem description facts for each lesson as being either required or remedial. As the student progresses through the required problems, any deficiency inferred by the diagnostic rules is recorded in the student model. On the basis of the student's performance, the system selects the next problem to present. If sufficient deficiencies have been recorded for a particular goal, a remedial problem is immediately presented to the student before the next required problem.

Clancey et al. [7] listed four major information sources for maintaining the student model: a) student performance progress observed by the system; b) direct questions asked of the student; c) assumptions based on the student's learning experience; and d) assumptions based on some measures of the difficulty of the subject matter material. The prototype CLIPSIT student model relies solely on observing past performance. A past performance history of the student's misconceptions is carried over to later lessons. This information is then used to select remedial problems when the student demonstrates misconceptions about goals presented in previous lessons.

USER INTERFACE

The design of the interface can make or break the effectiveness of a tutor, regardless of the clever design of the other components. If the user interface is confusing or non-supportive of the tutored domain, the effectiveness of the instruction will be diminished or lost entirely. A powerful user interface has been developed for CLIPSIT. The interface, written in C and utilizing a commercial graphics package, provides two windows for system text and user input. Window dimensions are controllable by both the tutor and student. Cursor keys or a mouse may be used to scroll the windows. Additional types of pop-up windows are available for menus, student responses, and diagnostic and help messages.

A series of functions are available which pass student responses to those portions of the tutor which handle error detection and instructional strategies. These input/output functions provide a general method for capturing student responses token-by-token, moving the cursor between the windows, permitting limited

student editing, writing text to windows, and special message formatting.

SUMMARY

The basic tradeoff which greatly affected the design of CLIPSIT was the issue of response time versus capabilities, that is, the ability of the tutor to respond to a student response fast enough for the student to avoid confusion and frustration versus the amount of flexibility and power that could be implemented. The intent was to create the most powerful teaching package available within a 640k memory constraint and with an average response time of 1-2 seconds. It was felt that a response time greater than 2 seconds between student responses would leave the student confused about what (if anything) was happening with the tutor rather than concentrating on the problem at hand. This fast response time was achieved and even surpassed using a PC/AT class of computer. Response times for the PC/XT class of computers usually met this constraint but would struggle once the student took a solution path much different from the one anticipated.

In order to minimize response time, the key strategy implemented was to represent as much knowledge about the problem as possible in the problem facts rather than make the system use many production rules to reach similar conclusions. For example, each problem file contains facts specifying how the problem should be presented to the student, that is, whether to use a template or not and how much help should be provided. This approach greatly simplifies the teaching expert function. The problem description facts also specify the possible variations to the solution that the student may make. These facts greatly simplify the domain expert.

One of CLIPSIT's greatest strengths is that all production rules are generic in nature. By eliminating problem specific rules, the size of the rule base decreases and typically execution speed improves. All problem specific facts are stored in separate problem files and loaded only when necessary. This minimizes the size of the fact base as well as the number of partial rule instantiations. An additional benefit of storing problem facts in an individual file and using generic rules is maintainability. Separate problem files can be generated by someone unfamiliar with CLIPSIT or rule based expert systems. Additional problems can be easily added, removed, or altered with no changes to the generic rule base.

REFERENCES

1. "CLIPS" is an acronym for "C Language Integrated Production System" and was developed by the Artificial Intelligence Section, Mail Code FM72, NASA/Johnson Space Center, Houston, Tx 77058.
2. Wallach, Brett "Development Strategies for ICAI on Small Computers," in Kearsley, G.P., ed., Artificial Intelligence and Instruction: Applications and Methods (Reading, MA : Addison Wesley Publishing Co. ,1987).
3. Brown, J.S., & Burton, R.R., "Diagnostic models for procedural bugs in basic mathematical skills," Cognitive Science, 2, 155-192.
4. Wenger, Etienne, "Basic Issues," Artificial Intelligence and Tutoring Systems, M. Morgan ,ed.,(Los Altos CA : Morgan Kaufmann Publishers, 1987).
5. Matz, M. "Towards a process model for high school algebra errors," in D. Sleeman & J.S. Brown,eds., Intelligent Tutoring Systems (New York : Academic Press 1982).
6. Carr, b., & Goldstein, I.P., "Overlays: A theory of modeling for computer aided instruction," Artificial Intelligence Laboratory Memo 406 (Logo Memo 40), Massachusettes Institute of Technology, 1977.
7. Clancey, W.J., Barnett, J.J., & Cohen, P.R., "Applications-oriented AI research:Education," in A. Barr & E.A. Feigenbaum, eds., The handbook on Artificial Intelligence (Vol 2) (Los Altos, CA : William Kaufmann, 1982).

SARSCEST
(Human Factors)

H. McIlvaine Parsons
Essex Corporation
Alexandria, VA

INTRODUCTION

As we all know, people interact with the processes and products of contemporary technology. Individuals are affected by these in various ways and individuals shape them. Some eager souls investigate such interactions or try to influence them, under the label "human factors."

The term is easier to define--as I have just done--than the considerable territory it labels is to describe. What is in it? To expand the understanding of those to whom it is relatively unfamiliar, its domain includes both an applied science and applications of knowledge. It means both research and development, with implications of research both for basic science and for development. It encompasses not only design and testing but also training and personnel requirements, even though some unwisely try to split these apart both by name and institutionally. The territory includes more than performance at work, though concentration on that aspect, epitomized in the derivation of the term ergonomics, has overshadowed human factors interests in interactions between technology and the home, health, safety, consumers, children and later life, the handicapped, sports and recreation, education, and travel.

Technology affects our emotions, but other than stress these, unfortunately, have aroused little interest, as though it were immaterial whether technology adds or subtracts enjoyment or depression or anxiety in our daily lives. Technology affects the interactions between individuals--our social and organizational lives, and these influence how we use or create technology. NASA has been alert to these aspects, which have been relatively neglected in human factors otherwise despite efforts to widen an involvement in studies of habitability and organization design. Who knows? Eventually human factors scientists and engineers may also investigate relationships between technology and motivational variables in the behavior of those who use its products and processes--especially incentives and disincentives, rewards and deterrents, and conditions that strengthen or weaken these.

But rather than talk about such matters, important as they are, in this paper I shall discuss two

aspects of technology I consider most significant for work performance, systems and automation, and four approaches to these that hold much current interest for both the Air Force and NASA and present new challenges for human factors science and applications.

SYSTEMS

This topic accounts for the initial S in the paper's title, which is an acronym. (Acronyms are a product of technology, in part, reflecting the need to conserve program space in software and the law of least effort in humans. Another instance is SOAR.) Human factors scientists and engineers deal with systems as well as components. A system model consists of input, processing, output, and feedback (SIPOF); the corresponding human model is HIPOF.

One contribution human factors can make is a systems perspective. This does more than include the roles of humans as well as machines in some proposed or developing system. It considers more than a single workplace for one or two individuals. It embraces more than optimizing the design; the human can also be improved. It extends beyond some traditional boundaries in investigating human-machine performance, such as servo-controlled negative feedback loops. It asks how the goals, set points, or criteria are established in task or mission planning. It views an automated or semi-automated system in more than a control context. Indeed, the human role in control may be virtually eliminated or is diminished. But the human roles in programming and maintenance are likely to become more significant. In fact, some maintenance operations, as in space travel, may be the primary human interventions--though I believe astronauts have not usually characterized themselves as "maintenance personnel." As a new AAMRL project demonstrates (Mohr, 1986; Julian and Anderson, 198), flight-line maintenance of aircraft is a candidate for robotic automation. The design of applications software--and training to use it--has become a major target of human factors research and applications. Maintenance and programming, along with planning and control operations, constitute interrelated system loops, so to speak, to be improved with help from human factors.

AUTOMATION

Thus heading supplies the second letter in the paper's acronymic title. Innovations in automation impose difficult demands on human factors. From the perspective of either a system or a workplace, what should be the division of labor? What should be automated and what left to some human performance? This is the proverbial allocation-of-tasks-and-functions problem, perhaps better stated as a combination-of-tasks (or subtasks) problem to emphasize what is properly its synergistic or symbiotic nature. It is novel to neither the Air Force nor NASA, as, for example, I found in a NAS-AF summer study for the former in 1981 (Air Force Studies Board, 1982) and a 1984 workshop on teleoperation for the latter. Nor is it novel in other contexts. One outstanding instance is machine translation, heralded in the early 1960s by the youthful artificial intelligentsia as a great advance in automation. When I spent a summer in Vietnam in 1970 for our Department of Defense to try to improve the training of the ROV armed forces, I was assured by our military personnel there that machine translation was about to solve the problem of converting masses of technical documents from English to Vietnamese. Five years later as a consultant to the United Nations I encountered wishful thinking that simultaneous interpretation in the General Assembly and Security Council could be automated in short order. Recently I found that the Pan-American Health Organization was among those finally using machine translation--but with post-editing by a human translator operating a word processor. Why had it taken so long to adopt the symbiotic solution rather than cling to the dichotomous position that humans and automation shouldn't mix?

Such experiences have left a residue of skepticism, modified, to be sure, by the automation that is assisting me in typing this paper. (On occasion I still use my old mechanical typewriter, however, especially during power failures when it's the only contrivance in my company's offices with which to compose some urgent communication.) The dilemma for human factors professionals springs not from any resistance to automation. In a world much of which could hardly exist if the automobile had not replaced the horse, it is essential. Nor is it likely that automation will put us human factors folks out of work; the more complex technology becomes due to automation, the more we seem to be needed. Rather, we don't know what to believe. If prognostications about new automation reflect wishful thinking and hyperbole, the historical record shows that advances in automation will nevertheless occur while the general competence of people remains in a steady state (except for variations due to training). But what, exactly, will be the advances, and how soon? Fortunately, computer scientists and electronic and mechanical engineers are less skeptical; they rush in where angels fear to tread. Unfortunately, they usually disregard the angels--who, of course, are the human factors scientists and engineers.

The division of labor is often determined before a human factors analyst has a chance to propose one. If the opportunity does occur, various guidelines can be invoked, including the relative competencies of automation and personnel, their relative costs, and the

prospects of improved (human-engineered) design, training, and personnel selection as alternatives. A symbiotic strategy may be proposed at least for the interim before new automation is perfected, perhaps with the rationale that the system developer will regress to that strategy in any case as the hyperbole melts in the heat of reality.

In this paper the four kinds of technology discussed have differing distributions of labor. They will be presented in a crude rank order from least to most human involvement. They are robotics, supervisory control, expert systems, and telepresence (which includes telerobotics and teleoperators). Together their initials generate the remaining letters of SARSCEST. At the end of the paper the four will be related to each other.

ROBOTICS

Non-technical observers consider industrial robots as autonomous because the observers rarely if ever see them being maintained or programmed. Nor do they usually see an operator start or stop them, calibrate them, intervene when some tool or workpiece needs adjustment, or change their programs; in any case, compared with closed-loop human control of machines such operations are relatively simple and infrequent. But robotic maintenance, either preventive or remedial, is a fact of life on the factory floor, as might be expected for mechanical, electrical, electronic, and hydraulic machines that incur wear from use and interference from ambient conditions. Such maintenance has human factors interest to the same extent that machines with similar components arouse such interest, except that industrial robots present special hazards due to unexpected movements.

It is the creation and use of application programs that call particularly for human factors investigation and application, as I discovered in a survey I conducted among ten major robot manufacturers for the Army's Human Engineering Laboratory (Parsons and Mavor, 1986). The following is taken from a recent summary (Parsons, 1988a; see also Parsons, 1988b,c) dealing with industrial robots and their programming.

The principal equipment units are a teach pendant (portable control panel) and a computer terminal; for programming continuous paths, as in spray painting, controls are attached to the robot arm or to a surrogate arm. A major design problem has been the division of programming between the pendant and the terminal; the former is carried inside the robot work envelope (bounded by a barrier or fence, with a gate), whereas the latter is usually next to the robot controller housing a microprocessor outside the enclosure. To program with sufficient precision the locations to which the robot's end effector will move automatically in a production run, in some tasks the programmer must go very close to the alignment of the end effector with a workpiece, operating the pendant's controls to move the robot. This requirement for visual discrimination accounts largely for the portable control panel. But many additional steps have to be programmed, such as robot arm speeds, pauses, inputs from and outputs to other equipment, decisions, and

repetitions. The hardware/software designer must choose between adding control and display components to the pendant, thereby increasing its size and weight, and performing additional programming at the terminal--rushing in and out of the work envelope to do so.

Teach Pendant Design

Presumably improving the pendant's design should be a human factors objective. Its controls must be operated quickly and without error, to save production time and forestall errors that might result in accidents or interruptions. Since exclusive space is not available for all the required functions, more than one of these may be assigned to a single control element, with selection by another element. What functions are assigned, and how? How are control elements grouped and coded? Which are better, push buttons or joysticks? What should appear on the LCD display? Might the pendant use a menu system with soft buttons? How should errors be indicated? What warnings are needed? Though the dozens of functions on the pendant make it seem like a simple enough device relative to the much larger number on control and display panels in a nuclear power control room, nevertheless the pendant design is a challenge.

That challenge has been met, according to the survey mentioned, by a great variety of designs. No two pendants are alike. There has been virtually no systematic resort to human factors applications. The same can be said of the associated applications software, as in the names and categorizations of commands, the structures of menus, and error prevention and recovery. Neither the hardware nor software appears to be as user-friendly as it might be. Indeed, as a result of the survey the Robotic Industries Association is attempting to generate an ANSI standard for the human engineering design of teach pendants, though to date this has consisted mostly of copying sections from MIL-STD-1472C.

But are teach pendants really that important for programming industrial robots? What alternatives are there to this use of the robot as its own measurement device, so to speak? If robotic applications programming could be entirely textual, computing positional and rotational information based on data from other sources, there would be no need for pendants. Such sources include machine vision and other sensors, stored geometric and other data (world models) about robots and workplaces (for example, from CAD), and graphical simulation. Indeed, it has been suggested that these will reach a level of precision or flexibility to cope with the variability that occurs even in an engineered environment such as a robot installation. For some installations, involving assembly tasks by small robots, teach pendants seem unnecessary. The basic questions are how reliable other data sources can be made and how much variability and unpredictability will be encountered. These are even more serious in other environments, such as in space and especially in military settings where there may be variability and unpredictability due to an adversary. Such variability and unpredictability range along a continuum. "Structured" and "unstructured" environments are a false dichotomy.

Meanwhile, as other data sources improve for programming industrial robots, one human factors task may be that of helping to fuse data from various sources, including teach pendants--which may function primarily to post-edit the position and orientation commands otherwise derived.

When it is more widely realized that "autonomous" robots must be programmed and maintained--at some cost and by human skills--there may be less optimism about the tasks they may undertake outside of manufacturing, especially in homes, in service occupations, and in space, and especially for occasional rather than repetitive tasks for which remote or removed human operators or supervisors are available. Again, much will depend on the versatility of the robot and the predictability of the setting.

SUPERVISORY CONTROL

In the present context, the term "supervisory control" labels one solution to the issue of allocating or combining tasks among humans and machines. As we all are aware, supervisory control exists also among humans. If I may be permitted a personal reflection, a single individual can have experienced it as a child supervised by parents (and vice versa), driving a car with back-seat advice (also vice versa), as a student controlled by teachers (again vice versa), as a newspaper reporter subject to the whims of editors, as a ship's commander exerting military control, as an investigator supervising the behavior of white rats, and as a R&D manager trying to control the performance of investigators. The varieties of situations and types of supervision in supervisory control among humans are paralleled in those among humans and machines.

Basically, it seems that the term "supervisory control" in human-machine systems has been introduced to denote one or more humans presumably at a computer linked to one or more computers directly controlling some process or device(s) in real time and acquiring information that is sent to the supervisory controller. Thus, the human role, except for interventions when there's a special problem or crisis, does not include much real-time control--and the term "supervisory control" is something of a misnomer. According to a report of the Committee on Human Factors of the National Academy of Sciences and National Research Council (Sheridan and Hennessy, 1984), "supervisory control behavior is interpreted to apply broadly to vehicle control (aircraft and spacecraft, ships undersea vehicles), continuous process control (oil, chemicals, power generation), and robots and discrete task machines (manufacturing, space, undersea mining)" (p.8), and also (p. 5) command, control and communication systems. I believe the term originated from human factors work on undersea and space teleoperations and on nuclear power plant control rooms. This kind of division of labor has been aimed at helping to help solve communication and workload problems. For example, supervisory control might cope with the problem of communication lags between NASA ground control and distant teleoperated space vehicles and manipulators. It might ease the complex and time-limited monitoring and remediation operations in nuclear power control rooms.

EXPERT SYSTEMS

According to Sheridan and Hennessy (1984, p. 12), "In all forms of supervisory control there is a typical five-step cycle in the human supervisor's behavior: (1) planning, including the setting of subgoals relative to the given task goals, (2) instructing the computer, (3) monitoring its execution of instructions and making minor adjustments, (4) intervening to circumvent the automatic controller as necessary, and (5) learning from the experience in order to plan better." In the proceedings of a symposium on human factors in automated and robotic space systems, Sheridan (1987, p. 286) observed that "The problem of what to control manually and what to have the computer execute by following supervisory instructions is something that cannot be solved in general but probably must be decided in each new context."

Akin, Howard, and Oliveira (1983) distinguished two types of supervisory control: traded and shared. In the former, the human defines a subtask for a computer, which then takes over, the human interrupting at will. In the latter, the human describes goals and manipulates high level information from a world model or textual input, and the on-site computer modifies the human's commands if there are misjudgments or delays, using a more precise world model.

Some lessons from all-human parallels seem pertinent. The human subordinate is often given considerable responsibility and authority so he or she can learn by doing, including mistakes, except perhaps in a real crisis. If the subordinate in human-machine supervisory control is a computer, it too must be adaptable. That means that the subordinate computer must acquire new behavior through feedback from its own performance and built-in learning algorithms or heuristics or by being tutored by the supervisory controller, perhaps through simulation. Another lesson is that supervision may extend through a hierarchy or be exerted by or on multiple individuals. Hierarchical aspects have been noted by the sources already cited, as well as control over multiple subprocesses--limits on human span of control being one reason for computer automation. The third lesson is that the supervisory relationship might in some situations be reversed, with a human at the "subordinate" position exercising considerable choice and a computer at the "supervisory" position developing plans and issuing them without human participation. The tail wags the dog, for example, when to avoid a major, unexpected obstacle the remote driver of a personless vehicle changes a supervisory computer's entire routing--perhaps an instance of shared control.

"Supervisory control" seems to need more analysis from actual systems and empirical support as to its applicability beyond laboratory studies that launched the concept. Although its descriptions have not given much heed to maintenance and programming, perhaps because of its explicit emphasis on "control," some supervisory control interventions, as in nuclear power incidents, would actually appear to concern maintenance, that is, troubleshooting and remediation. Suggestions that some programming is supervisory control will be noted subsequently.

An expert system is an artificial intelligence undertaking incorporating an interactive computer program containing expertise for a non-expert end user. Why should such a system involve human interfaces and thus human factors? First, though it manipulates words (or graphics) instead of sensor inputs and motor outputs, it possesses the same SIPOF structure as other types of systems to which human factors science and practice are applicable (though the F--feedback--is not explicitly considered by AI developers in this venture any more than in others). Second, it has three or more human interfaces, more than enough to justify human factors interventions. Third, it represents, in a way, the apotheosis of symbiosis in computer automation.

Although human factors support apparently was not sought by ES developers initially (and ESs were more processing-oriented than user-oriented, perhaps to their disadvantage), the human factors community has leaped into the act. The Proceedings of the annual meetings of the Human Factors Society had two ES presentations in 1984, three in 1985, eight in 1986, and six in 1987. Of the 19, four discussed human factors in ES in general, 11 were concerned with expert systems in particular contexts--four of these being aircraft and seven troubleshooting and other search tasks, and four described the use of ES as human factors methodology. Among these last, Antonelli (1987) compared an ES with a hard copy user's manual as a job performance aid; Parng and Ellingstad (1987) developed an ES to help design menu systems; Hartley and Rice (1987) developed an ES to help design the screen color of video displays; and Karwowski, Mulholland, and Ward (1986) developed an ES to analyze manual lifting tasks. The aircraft-related ESs were authored by Endsley (1987), Kuperman and Wilson (1986), Aretz et al. (1986), and Ott (1985); two originated in the Armstrong Aerospace Medical Research Laboratory and a third in the Air Force Academy. Of the troubleshooting/search ES talks, two arose from work for NASA at Carlow Associates (Phillips et al., 1986; Eike et al., 1986); two from work for the Electric Power Research Institute (EPRI) at Honeywell (Koch, 1985, 1987); one from Westinghouse (Roth et al., 1985); one from Idaho National Engineering Laboratory (Nelson, 1986); and one from Martin Marietta--Denver (Petersen et al., 1984). The general discussions were by Hamill (1984), Ostberg (1986), Fotta (1986) and Gehlen and Schwartz (1987), this last a study of display formatting. Other examinations of expert systems from a human factors viewpoint have appeared in journals.

AI descriptions of expert systems conventionally specify three parts: knowledge engineering to create a knowledge base, an inference engine to manipulate this, and a user interface. A SIPOF model, followed here, differs somewhat. A system component (S) is where decisions are made as to what domain should be put in an ES or left to other, perhaps non-computer, methods. Another decision is whether the ES should be "autonomous" or advisory; some ESs are the former. The second, input (I) component is knowledge acquisition (facts, relations, well or ill defined, in numeric or symbolic form). It has two parts. One is the expert

source--a human expert providing self-reports through interviews or questionnaires, manuals or other literature, think-through protocols, simulations/scenarios, even experiments--though these are mentioned as rarely in this AI context as in others. The second part is the agent acquiring the knowledge, the knowledge engineer in the conventional ES description. In the third, two-part processing (P) component, the acquired knowledge is first converted (by a knowledge engineer) into a form that can be computer-processed. It is decomposed or structured into rules, frames, scripts, schemas, arrays, property lists, hierarchies, semantic networks, analogical representations, or propositional or predicate calculus--relational formalisms on which the program in the inference engine can operate; these may be essentially procedural in form ("how" sequences) or situational/declarative, perhaps concerning non-verbal skills. The conversion process must attempt to standardize terms and make them consistent. It may include criteria for the selection of the premises in (production) rules. Confidence/certainty values can be assigned to items as well as importance values. In the second part of processing, the resulting "knowledge base" is further structured and sequenced in the computer program so the program in interaction with a user can produce various relationships, e.g., causal or hierarchical, with combinations of premises and forward or backward sequencing among rules related by sharing premises or conclusions. The fourth (Output--O) component is the interface with the user. It consists of presentations or queries (by computer or user) on a visual display (or perhaps by voice synthesis and recognition), proceeding through the program sequences, culminating with a conclusion or recommendations (advice). This O component also usually includes explanations of the program's reasoning process and conclusions, i.e., rationales. It may note conflicts and trade-offs. Some ESs include lists of references for further information or as evidence. The last (feedback--F) component in an ES, though seldom if ever identified as such, is implied in user queries, user additions to or elaborations of the knowledge base, and subsequent growth of the ES through iterations. In addition, a human factors approach calls for examining interactions between these components and for conducting a systematic ES evaluation.

The three human interfaces are (1) between the source (if a human expert) and the knowledge engineer; (2) between the knowledge engineer and the inference engine; and (3) between the output of the inference engine and the user. Some interfacing occurs also between the inference engine and the user or source expert.

Some Human Factors Considerations

System. As already noted, a determination must be made whether the ES's conclusions will simply inform the user or be advisory--an allocation matter. Criteria include criticality, frequency, technological capabilities, and pilot acceptance/trust (Endsley, 1987). Should the simpler or more routine tasks receive priority for ES handling, though some may be trivial? The same author said that expert systems are "severely limited in dealing with unforeseen circumstances," another consideration typical of computer-based systems.

Input. How large a domain should be included? It has been generally accepted that expert systems can deal with only limited domains. How complete is the input even within a limited domain? According to Ostberg (1986), an ES is likely to omit tacit or "common sense" knowledge. What types of domains or repertoires are appropriate? What sorts of "thought" can be verbalized reliably or at all by an expert--though much of the thought may have been originally verbal? Expert systems are inevitably subject to biases, misinformation, inadvertent or deliberate misinterpretations or exaggerations, omissions, rationalizations, dependent as they are on human long-term memories, mismatches in sophistication between expert and knowledge elicitor, and the latter's incentives and disincentives. Conceivably during the current national election campaign each major political party might produce an expert system demonstrating not only that it should win but would win--especially in light of the diversity of views of economists or columnists who might be the "experts." Also possible but perhaps less likely is the exploitation of expert systems by UFOlogists, astrologers, and others victimizing the credulous with occult claims. ES dependence on self-report induces caution in any human factors scientist or engineer, to whom the hazards of this kind of evidence or knowledge are well known. One way around this dilemma is to query more than one expert (Endsley, 1987). But an extensive survey of USAF jet interceptor pilots showed considerable disagreement among them about the procedures they used and how a return-to-base system should be designed (Parsons and Karroll, 1954). Should expert views be determined by majority vote? Much work apparently remains to be done to assure suitable and reliable input to ESs.

Processing. In the initial processing by a "knowledge engineer" there lie further hazards. That individual may have many options concerning the techniques to invoke, formalisms to use, terms to select, and confidence or importance values to attach. According to Ostberg (1986), knowledge engineers constitute a "priesthood" or can be viewed as "artists" with unsupported pretensions. Though they are analogous to work-study engineers or human factors engineers conducting task analyses, it is not clear what formal training they receive or should receive. How well can they orient to the needs of users as well as experts, especially if their cognitive styles differ? Categorization differs widely among individuals, as do direction of inference (data-driven or hypothesis-driven) and goals or incentives/disincentives.

In the processing in the ES computer program there may lie a mismatch between what the user is accustomed to do in his or her head and what the program does. If the user insists on knowing what the program has been doing, as astronauts and pilots, for example, may well do, the user may have difficulty in understanding or may not accept the conclusion or recommendation. Logic manipulations may perplex. Just because they are symbolic does not make them so different from perceptual-motor performance which, for example, requires designs that conform to group stereotypes. Mental (verbal, in-the-head) models vary among individuals as do perceptual-motor models. Users can misinfer, much as an operator may move a

control element in the wrong direction because the designer has designed it without due regard to the operator's habitual performance. As already indicated, mental models may vary in the direction of inference, data-driven forward chaining or hypothesis-driven backward chaining--which are alternatives for inference engines.

Output. According to Ostberg (1986), the user interface accounts for almost one-half of ES programming but has often been poorly done. Perhaps the first matter to consider is a user's goals, which Pew (1988) has emphasized. Since the user wants to put the output to some end, what kinds of purposes exist? Expert systems are aimed at decision-aiding and problem-solving of various kinds, predictions as well as comprehension. They are intended to do what the non-expert cannot do, or do well, and should be tailored to the user's goals, not what a program can accomplish. They should also be tailored to users' levels of expertise, which will vary among them. Aretz et al. (1986) advocated designing in three levels, Parng et al. (1987) six. So the knowledge engineer or the computer program must be flexible and function as a kind of translator between the expert and the user. When an ES is used for human factors purposes, these as we have seen can include trouble-shooting, replacement of TPAs, design of hardware or software, personnel selection, and training (as in coaching new technicians). These objectives may call for different interface aspects. A major output consideration is the kind of language that is desirable. Although this has been described as "natural," that term calls for better definition; everyday spoken language not only differs from the written but is likely to vary widely in syntax or a lack thereof. Some restrictions seem inevitable to avoid confusion, since users will have their own vocabularies and usages. That fundamental human proclivity, generalization, is a major problem (as well as an advantage) in symbolic communication, though it has not always been recognized as such under this label. (Concepts, heuristics, and synonyms are examples of generalization, and fuzzy set theory has tried to cope with it in a limited fashion.) Along this line, Pew (1988) has noted the variations in users' abstraction levels. Because some ES output may be difficult to convey in symbolic form, it may be advisable to use graphic representations.

A major aspect of ES output is the capability it includes for explaining to the user how or why it reached some conclusion or recommendation. As Pew (1988) and others have noted, apparently it is not sufficient simply to list the rules the inference engine used or to convert these into text. Citing other sources, Eike et al. (1986) listed five criteria for an explanation facility in an ES. For example, an ES "should be able to alert the user when a problem is beyond its current capabilities and instruct the user as to what additional factors and/or rules would be required to complete the transaction." Further, "the explanation facility should be capable of recalling each invoked rule and associating it with a specific event to explain the rationale for the machine's assessment of the event."

Feedback. Despite the rarity of specific mention in the AI literature, some feedback does occur in the

use of expert systems. For example, a user's queries about methods the ES employed, requests for justification, and meaning of questions the program asked (Phillips et al., 1986) can be regarded as kinds of feedback to the ES. Still another variety would be program or knowledge base updating or alteration requested by the user. Perhaps the most significant feedback is rejection of the expert system by intended users. Such may occur because they find it too difficult, they get lost, or they lose confidence in it. Or the system, especially if the user must query it for information or explanation, increases rather than diminishes the workload--an especially sensitive matter for military pilots. In short, due to the consequences of using an ES, an individual avoids doing so, in what would technically be called avoidance conditioning.

Evaluation

The need for evaluating expert systems before their use seems self-evident but has not always been satisfied; in fact, evaluation appears to have occurred mostly in the course of their use, on occasion with poor results. But Kuperman and Wilson (1986) have described a research facility to include ES assessment at AAMRL. Aretz et al. (1986), Nelson (1986) and Koch (1987) have reported experimental evaluations. Ott (1985) listed ES programs within the Department of Defense and NASA. Though he did not include individual assessments--which may not have been available, he did contrast current and desired/required capabilities of "intelligent aiding in the cockpit" and concluded that "significant additional progress must be made before an AI system can be used in a combat airborne application."

TELEPRESENCE

By and large, manipulators and vehicles are remotely operated (or maintained) by humans (1) when it would be too hazardous or, possibly, too arduous, for a human to be in the same place or (2) the nature of the task (e.g., non-repetitive or too unpredictable) argues against using an autonomous robot in view of the robot's total cost (including programming) and lower level of adaptability. But like human activities in general, remote manipulation and locomotion depend on sensory feedback to the human operator. How much and of what kind are important and interesting human factors issues.

The increasing use of teleoperators has given prominence to information feedback's significance in human performance (Smith and Smith, 1985). In general, people are not nearly as aware of the feedback they receive, as feedback, as they are of sensory inputs that do not seem contingent on what they do. There seem to be some kinds of information feedback of which we are especially unaware, perhaps because we cannot see the receptors. These are feedbacks from movements and positions of parts of our body to kinesthetic or proprioceptive receptors conveying information about movement and positions in space. Unlike vision and audition, kinesthetic/proprioceptive feedback is difficult to isolate and manipulate experimentally, so it has been studied less than these other

senses in investigations of human performance. Delayed feedback in audition in the past has dramatized feedback's role in audition, and delayed feedback in teleoperation has more recently emphasized its significance in kinesthesia/proprioception.

To some this is a welcome development. Feedback in general has received short shrift in cognitive psychology, notably the kinesthetic/proprioceptive variety, which occurs outside the cranium and seems foreign to self-report. But teleoperations' dependence on feedback through a number of sensory modalities forces a more comprehensive understanding of our behavior.

If I were compelled to select an alternative term for teleoperations or telerobotics, I might choose telefeedback. But I have been preempted by the choice of "telepresence," and I'm sufficiently grateful for any label that emphasizes in any way the need for examining sensory feedback that I won't quibble.

"Telepresence" calls attention also to the significance of generalization, another piece of behavioral bedrock (Shepard, 1987). Essentially, telepresence means the generalization of sensory inputs from those that would occur if an operator were at the remote manipulator or vehicle to those where the operator is actually present. The aim is to enhance such generalization-- which also has been called transfer, stimulus equivalence, and metaphor. Generalization, whether of input or output (acts), has likewise received relatively short shrift in cognitive psychology, perhaps because of its behaviorist forebears. However, it is the behavioral basis of such concepts as fidelity, validity, realism, and verisimilitude in simulation for training or experimentation and also in the super-simulation known as "virtual" imagery or proprioception and "artificial reality."

Telepresence, then, is linked to such concepts through generalization, as it is to empathy and *deja vu*. By increasing the fidelity and number of sensory channels, combining vision and proprioception and audition, it has effects on its audience--the teleoperation operators--not unlike what happened when color and sound were introduced into black and white and silent movies, television proved more appealing than radio, and, perhaps, video-conferencing has supplanted telephonic conference calls. Artists, novelists, poets, dramatists all have tried at times to make readers, viewers, or listeners feel they were in the situation depicted in words, music, or pictures, that is, to react in some fashion as though they were there.

Sense of Presence

But what is that reaction? It has been called "a sense of presence," an awareness, a perception, a "sense of being there." But physiologists would disclaim the existence of any sensory receptors that could be readily identified with these terms. Is "telepresence" poetry or is it science? Sheridan (1987) said that telepresence itself is not the goal of "telepresence"--it is really performance. But, he added, we should develop a cognitive theory of presence. He further (p. 279) discussed the "ideal" of sensing and communicating to a remote operator "in a sufficiently

natural way that she feels herself to be physically present at the remote site," and more restrictively, a teleoperator's dexterity matching a bare-handed operator's. Akin, Howard, and Oliveira (1983, p.35) equated telepresence with teleoperation that "makes the operator feel natural." Stark (1987, p. 298) said one should "provide the human operator with a telepresence feeling that he is actually in the remote site and controls the telemanipulator directly." The glossary in *Autonomy and the Human Element in Space* (1985), the final report of the 1983 NASA/ASEE Summer Faculty Workshop, gives the most comprehensive definition of telepresence: "Teleoperation with maximum sensory feedback to the operator, providing a feeling of 'being there' thus allowing greater precision and reliability in performance; remote presence."

The issue is whether better performance results from the feeling or performance and reported subjective reactions are merely correlated. Some justifications of telepresence seem to disregard performance, at least directly. The Soviets designed their cosmonauts' environments in part to resemble their home settings, so they would feel at home (Wise, 1988). As I suggested at the start of this paper, there's more to be investigated in human factors than simply work performance. In any case, the construct "telepresence" might benefit from experimental analysis and objective (or even subjective) measurement. For example, operators might be stationed at both the control and remote sites and their performances, feedbacks, and reported feelings recorded and compared.

If telepresence (as defined) is important, might it be augmented beyond three-dimensional CCTV, pressure or force sensing that mimics what is encountered by a teleoperator's hand in pushing or grasping or twisting, and beyond feedback from head and eye movements and gestures? What about feedback from facial and jaw tensions or movements and the musculature of the vocal apparatus? What about walking, stooping, climbing, and kneeling--or lying down, for that matter, asleep or awake? What about telepresence for locomotion? The term has been directed primarily at remote manipulation, but teleoperators can also be vehicular (perhaps carrying a manipulator, or only sensors, or nothing--e.g., for deception). Legged vehicles offer an obvious challenge to telepresence, especially since they have more legs than the human operator's. A wheeled or tracked vehicle is even more of a challenge, unless some cognitive link is inferred through the operator's familiarity with automobiles. If so, should the operator's controls resemble the steering wheel and pedals of his auto?

To what extent should there be concern about divergences between the teleoperator and individuals controlling it-- individuals who differ among themselves? Might divergences in forces exerted differ in scale--as they often do with teleoperators--thus improving performance but reducing the resemblance? How might reduced gravity or other divergent ambient effects be handled? What about rotational differences in joints, as well as differences in joint types, e.g., prismatic joints in telerobots that humans lack? To what extent should reactions be elicited from an operator such as he or she would experience if present at the teleoperator, such as anxieties, stress, fatigue,

excitement, even pleasure, so the operator will really feel as the operator would if he or she were there?

Performance

Though this discussion of telepresence has centered on subjective reactions in remote control, most human factors research and development in tele-operations has emphasized performance, whatever the label. Indeed, neither the Department of Energy nor the Department of the Army has been using the term "telepresence." The former has active human factors involvements in teleoperation research and development at Oak Ridge National Laboratory and at Sandia National Laboratory, the latter at its Human Engineering Laboratory at the Aberdeen Proving Ground (and elsewhere, including some work for it both at Oak Ridge and Sandia). The Navy's human factors work in teleoperations has been concentrated largely at the Naval Ocean Systems Center, Hawaii. NASA has for some time supported human factors in teleoperations (especially at the Jet Propulsion Laboratory and Ames Research Center), as have the Electric Power Research Institute, Woods Hole Oceanographic Institution, various universities, notably M.I.T., and several corporations, such as Martin Marietta, Lockheed, Grumman, ARD, and Essex.

To describe the work already done in these locations and continuing there would overextend the reach of this paper. It is fitting, however, to mention some recent projects. For example, Sandia has been developing the control station for the Army's TMAP (Teleoperated Mobile Antiarmor Platform) Project. Miller (1987) has reported a Sandia pilot study in which actual (interactive) and video-taped (simulated) remote driving were compared for search, detection, and clearance judgments of obstacles; the aim was to find out whether the video-taped technique could be used for more economical and better-controlled studies of variables in remote control of vehicles, such as TMAP, in difficult, off-road terrain. Sandia is interested in color vs. black-and-white video for viewing such terrain, field of view with panning vs. multiple cameras for local area navigation and turns, camera placement and steering coupling, and resolution effects. McGovern (1988) has described problems encountered in testing some of Sandia's fleet of seven remotely or directly controlled vehicles. These problems have included accidents due to tilt and roll, non-avoidance of "negative" obstacles (e.g., holes), over-control in steering, television minification vs. magnification, and recognition of spatial landmarks.

According to Spain (1987), NOSC has developed six remote-driving courses with traffic cones to investigate different aspects of low-speed maneuverability under four conditions: direct (on-board) driver viewing, partially masked direct viewing, on-board viewing through stereo TV with the same field of view, and remote stereo TV viewing. NOSC is examining remote control of the Marine Corps TOV (TeleOperated Vehicle) using an Army HMMWV (High Mobility Multi-purpose Vehicle). It plans to study two new stereo TV display systems. In one, the right eye sees a full 60 by 45 degrees view while the left sees a higher-resolution central area of 20 by 15 degrees. The other has a retroreflective screen and beam splitter to provide a

wide field of view. In a review of past, current, and future NOSC work, Hightower, Smith, and Wiker (1986) described a "hybrid" stereo display system in which one eye can get a high resolution black-and-white image and the other eye one in color. According to these authors, "NOSC has joined in a cooperative research and development effort with NASA to develop principles for design and construction of tactile display systems." One project involves a computer-graphics (virtual) arm.

At NASA's Ames Research Center, "an interactive virtual environment display system controlled by operator position, voice and gesture" has been developed in the Aerospace Human Factors Research Division (Fisher et al., 1987). It incorporates a wide-angle helmet-mounted stereo display that tracks head movement, lightweight glove-like devices that transmit finger shapes and hand and arm positions and orientations (gestures), connected speech recognition and synthesis in 3D, and 3D-graphic virtual objects with an articulated hand. Fisher (1986) has described the glove-like device ("DataGlove") in detail. A number of other simulated hands have been developed recently (Leifer, 1987; Thiele, 1987; and others).

My own organization, Essex Corporation, has done extensive human factors investigations in tele-operations for NASA at Marshall Space Flight Center and more recently for the Army's Human Engineering Laboratory. Currently we are helping H.E.L. develop a robotics laboratory as part of its Soldier-Robot Interface Program. Our responsibility is the human interface portion with associated software and equipment in a versatile control center; other organizations are providing the vehicle, a manipulator and sensors on the vehicle, and the software with which ours will interface through a fiber optics link. This facility will enable H.E.L. to experiment with different configurations and components of display and control equipment in a range of variables, such as 3D television and various types of controls. The control center will have both an operator station and an experimenter's station for both test direction and data collection, the former accommodating two operators so various dual control procedures can be investigated. Perhaps the most challenging aspect of this project is that it includes the control of both a vehicle and a manipulator, as well as sensors for each. For the most part, other human factors projects have concentrated on either the manipulator or the vehicle, with sensors for one or the other.

At an International Symposium on Teleoperation and Control just concluded at the University of Bristol, England, papers were given by A.K. Bejezy and B. Hannaford (JPL) on "Man-Machine Interaction in Space Telerobotics," R.L. Pepper (NOSC) on "Telepresence and Performance Assessment," C. Blais and R. Lyons (General Dynamics) on "Telepresence: Enough Is Enough," J.V. Draper et al. (ORNL) on "High Definition Television: Evaluation for Remote Task Performance," B.K. Lindauer and C.S. Hartley (Martin Marietta) on "Evolving Strategies for Supervised Autonomous Control," and a number of Europeans as well as several Americans who were referenced earlier and H.B. Meieran, who discussed tele-operations at the Chernobyl power site.

RELATIONSHIPS

To my knowledge, R, SC, ES, and T have not all been compared in any systematic analysis. What follows are some illustrations of how they relate to each other.

Robotics and Supervisory Control

When used with remote manipulators or personless vehicles, supervisory control occupies a position on the automation continuum between autonomous robots and telepresence (teleoperators). The actual control of the manipulator or vehicle resides in a co-located computer, as would be the case with an industrial robot, but a human can intervene and take over control, as in the case of a teleoperator. However, the human must generally intervene by means of another computer at the supervisory location.

Supervisory control can occur in a factory where a supervisor with a computer, or a computer with a supervisor, manages some number of devices, including robots, through their computers, in an automation network. In this situation, supervisory control primarily plans and schedules operations. Presumably its interventions consist mostly of starting, stopping, and interconnecting the robots and other devices in the network.

Most autonomous, factory robots are still teach-programmed, that is, a technician or engineer programs the robot's motions and other actions with a teach pendant or teaching arm or arm attachment. Akin, Howard, and Oliveira (1983, p. 58) described this sort of teach programming as a kind of "traded" supervisory control of an underwater manipulator, though without referencing its use for industrial robots. Sheridan (1987, p. 285) seemed to suggest something similar for trajectory programming as supervisory control.

Graphics simulation may be used in both robotics and supervisory control, though for somewhat different purposes. In robotics it can be exploited for off-line programming. In supervisory control it can forecast motions of a vehicle or manipulator, an especially useful technique when time delays occur in feedback messages.

Robotics and Expert Systems

In teach-programming an industrial robot, the programmer may be the same individual who originally performed a similar task before the robot was installed. Such is particularly likely in the teach-programming of trajectory or continuous robot movements, as in spray painting, in contrast to single or discrete positionings and orientations. Considerable skill is required and the extended movements of the teaching arm or of a robot with a teaching attachment may have to be repeated a number of times before the proper path can be recorded. In a sense, the programmer is an "expert" from pre-robotic experience with the same kind of task.

Akin, Howard, and Oliveira (1983) have likened the rules found in an ES to the "equations of motion of

the manipulator arm and its interaction with the environment," apparently referring to the forward and inverse kinematics in the program controlling the arm's movements.

Both autonomous robots and expert systems are limited in their abilities to cope with circumstances unforeseen by their programmers. They cannot easily adapt to the unexpected, though the robot may have an advantage due to its sensors. The robot depends otherwise on data in a "world model" of itself and its setting and on data from teach programming. The expert system depends on what has been placed in its "knowledge base," containing data from an expert as teacher combined with actions by a knowledge engineer. Each might be quite helpless if faced by a clever adversary.

The robot is more similar to a human, with simulations of at least one jointed arm, a torso, and various sensors, including vision. The expert system is entirely cerebral and verbal, with symbols for input and output. On the other hand, the ES includes several significant human interfaces, not including maintenance, and the robot essentially only the applications programming interface, again not including maintenance. More human-machine symbiosis occurs in the expert system.

Robotics and Telepresence

These are at opposite ends of the automation continuum in this paper. But they have a greater affinity than is usually recognized. The operations for programming a robot with a teach pendant resemble those for controlling a teleoperator. According to Mitchell (1987, p. 108), one way to train a robot "might be to use a teleoperator to guide the robot through several uses of the tool," which would be the equivalent of teach programming (though the author does not reference this). Though most teach pendants have push buttons for rotating or extending the links in the manipulator's arm, and a teleoperator's links are more likely to be actuated by a joystick, at least one major robot manufacturer uses a joystick, and a teleoperator could be controlled by push buttons. Resolved motion (in which joints move together to create a direct path to a point) occurs in both teleoperator control and robot teach programming.

The design of the workplace and of objects to be manipulated or avoided needs to be enhanced for both autonomous robots and teleoperators, whether stationary or mobile. Such enhancement can benefit the robot's programmer and the teleoperator's operator by making their tasks less complex.

Both robots and telepresence systems require sensors, especially vision. Robots use machine vision. Teleoperators use human vision. But both rely on closed circuit television as the intermediary. (Neither is as good as direct human vision.) Little heed has been given, it appears, to combining televised machine vision with televised human vision for either robots or teleoperation, a potential innovation in symbiosis. Some of this already occurs, in a limited fashion, when programmers create templates for machine vision (using the templates already in their heads) in

exploiting the human's superior pattern recognition. It has been suggested (Parsons and Mavor, 1986) that the same television apparatus for machine vision might be used in teach-programming a robot to relieve the programmer of the need to get close to the end effector with the teach pendant.

Supervisory Control and Expert Systems

An expert system might be regarded as a form of supervisory control. Or an expert system might be designed for the human supervisor's computer to aid in problem solving and decisions.

Expert systems are envisioned that can operate in real time, as, for example, in aircraft piloting, but most are not time constrained. Supervisory control may function in real time, in partially real time (when there's a communications delay, for example), and for some tasks, e.g., planning, without time constraints.

Supervisory Control and Telepresence

To the extent that "presence" rather than performance is emphasized in telepresence, telepresence may not be needed for supervisory control, or at least needed much less than for direct control. Except during interventions, the supervisory controller is not receiving visual or kinesthetic/proprioceptive feedback from control actions. One issue is whether non-feedback visual monitoring would benefit from a "sense of presence."

One reason for having supervisory control is to be able to manage multiple manipulators or, even more, multiple vehicles. Providing a "sense of presence" derived from each of a number of locations and vehicles might be too confusing both for intervention feedback and for monitoring. This could be an interesting empirical problem to investigate.

With regard to the performance aspects of telepresence, it can be repeated that supervisory control resides between telepresence and robots on this paper's automation continuum. One purpose of supervisory control is to diminish an operator's workload by assigning most of teleoperations' control functions to a computer co-located with the manipulator or vehicle.

Expert Systems and Telepresence

One way in which these differ is with respect to the "sense of presence" aspect of telepresence. Because the expert system has no sensors or limbs, it gets no feedback except from its user, verbally. Any non-feedback "presence" would have to be expressed symbolically or in graphics, and kinesthetic feedback would be difficult to put into words. Much of the "reality" of the scene might be missing.

In terms of performance, an expert system and telepresence have some resemblance in that the user of each must progress from point to point, symbolically in one, by vehicle or manually in the other. This analogy is apt, however, only for forward, data-driven chaining in the expert system (except, perhaps, for a vehicle's return journey from the objective).

Expert systems and telepresence's "sense of presence" aspect share the need for a greater amount of evaluation. The performance aspects of both hold great interest for human factors in view of the extensive symbiosis and complex human interfaces in each.

Further Comparisons

R, SC, ES and T might be further compared with respect to the list of items in Figure 1, specifying human factors involvements in these four automation-related processes; the extent of involvement for each listed item for each of the four processes can be filled in for the particular system examined. In addition, each of the processes can be examined with respect to the components of the SIPOV model. For "system" that means task allocation/combination; for "input," symbolic or imagerial (graphic) in form; for "processing," control, maintenance, or programming; for "output," symbolic, imagerial, or motor; and for "feedback," informational or motivational.

This last category, motivational feedback, refers to the incentives and disincentives that influence so much of our performance, though they have not been emphasized in human factors R&D or in this paper. Motivational and informational feedback may share the same feedback events but are functionally distinct. A major incentive (prospective consequence) or positive reinforcer (past consequence) is money. Cost/benefit ratios affect whether managers or investors will adopt any of the four processes discussed. Disincentives or deterrents include difficulty, excessive mental workload, errors, and other unfriendly consequences (stressors) that end users experience with a process, inducing them to avoid or cease using it. These motivational feedback variables are pertinent to robotics, supervisory control, expert systems, and telepresence--that is, to all of SARSCEST.

REFERENCES

- Air Force Studies Board (1982). Automation in combat aircraft. Washington, DC: National Academy Press.
- Akin, D.L., Howard, R.D., and Oliveira, J.S. (1983). Human factors in telepresence. NASA-CE-173420. Cambridge, MA: Massachusetts Institute of Technology.
- Autonomy and the human element in space (1985). Final Report of the 1983 NASA/ASEE Summer Faculty Workshop. Washington, D.C.: National Aeronautics and Aerospace Administration.
- Antonelli, D.C. (1987). A comparison of hardcopy and expert system presentation of network operator assistance guides. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.
- Aretz, A., Guardino, A., Porterfield, T. and McClain, J. (1986). Expert system advice: How should it be given? In Proceedings of the Human Factors Society--30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Eike, D.R., Fleger, S.A., and Phillips, E.R. (1986). User interface design guidelines for expert troubleshooting systems. In Proceedings of the Human Factors Society--30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Endsley, M.R. (1987). The application of human factors to the development of expert systems for advanced cockpits. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.

Fisher, S.S. (1986). Telepresence master glove controller for dexterous robotic end-effectors. In SPIE Vol. 726 Intelligent Robots and Computer Vision.

Fisher, S., McGreevy, M., Humphries, J., and Robinett, W. (1987). In Proceedings of the International Topical Meeting on Remote Systems and Robotics in Hostile Environments. La Grange Park, IL: American Nuclear Society.

Fotta, M.E. (1986). Applications of artificial intelligence to improving the intention stage of the human-computer interaction. In Proceedings of the Human Factors Society--30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Gehlen, J.R. ((1987). Display formatting: An expert system application. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.

Hamill, B.W. (1984). Psychological issues in the design of expert systems. In Proceedings of the Human Factors Society--28th Annual Meeting. Santa Monica, CA: Human Factors Society.

Hartley, C.S. and Rice, J.R. (1987). A desktop expert system as a human factors work aid. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.

Hightower, J.D., Smith, D.C., and Wiker, S.F. (1986). Development of remote presence technology for teleoperator systems. Presentation at the 14th Meeting of UJNR/MFP, Bethesda, MD.

Julian, R.G. and Anderson, T.R. (1988). Robotic telepresence: Applications of human controlled robots in Air Force maintenance. In M. Ung (Ed.), Proceedings of the Society for Computer Simulation Multiconference on Aerospace Simulation III, SCS International Simulation Series, 19(2).

Karwowski, W., Mulholland, N.O., and Ward, T.L. (1986). A knowledge-based expert system for the analysis of risk of overexertion in manual lifting. In Proceedings of the Human Factors Society-- 30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Koch, C.G. (1985). User interface design for maintenance/trouble-shooting expert system. In Proceedings of the Human Factors Society--29th Annual Meeting. Santa Monica, CA: Human Factors Society.

Koch, C.G. (1987). Human factors evaluation of gas turbine expert system. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.

Kuperman, G.G. and Wilson, D.L. (1986). An expert system approach to workload reduction. In Proceedings of the Human Factors Society-- 30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Leifer, L.J. (1987). Design, development, and evaluation of Stanford/Ames Eva Prehensors. Progress Report for NASA. Stanford, CA: Center for Design Research, Stanford University.

McGovern, D.E. (1988). Human interfaces in remote driving. Sandia Report SAND88-0562. Albuquerque, NM: Sandia National Laboratories.

Miller, D.P. (1988). Evaluation of vision systems for teleoperated land vehicles. IEEE Control Systems Magazine, 8(3), 37-41.

Mitchell, T.M. (1987). AI systems in the space station. In T.B. Sheridan, D.R. Kruser, and S. Deutsch (Eds.), Human factors in automated and robotic space systems: Proceedings of a symposium. Washington, DC: Committee on Human Factors, National Research Council.

Mohr, G.C. (1986). Robotic telepresence. In Proceedings of the Human Factors Society--30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Nelson, W.R. (1986). Operator aids and expert systems in user computer interfaces. In Proceedings of the Human Factors Society-- 30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Ostberg, O. (1986). Expert systems in a social environment--Human factors concerns. In Proceedings of the Human Factors Society-- 30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Ott, L.M. Jr. (1985). Artificial intelligence--What does it offer combat aircraft? In Proceedings of the Human Factors Society 29th Annual Meeting. Santa Monica, CA: Human Factors Society.

Parng, K.A. and Ellingstad, V.S. (1987). Menuda: A knowledge-based menu design expert system. In Proceedings of the Human Factors Society--31st Annual Meeting. Santa Monica, CA: Human Factors Society.

Parsons, H.M. (1988a). Human factors. In R.C. Dorf (Ed.), International encyclopedia of robotics: Applications and automation. New York: Wiley.

Parsons, H.M. (1988b). Robot programming. In M. Helander (Ed.), Handbook of human-computer interaction. Amsterdam: North-Holland.

Parsons, H.M. (1988c). Human factors in robot design and robotics. In D.J. Osborne (Ed.), International Reviews of Ergonomics--Volume 2. London: Taylor & Francis.

Parsons, H.M. and Karroll, J.E. (1954). Return-to-base questionnaire survey of jet interceptor pilots. Technical Report T-2/A-VII. New York: Electronics Research Laboratories, Columbia University.

Parsons, H.M. and Mavor, A.S. (1986). Human-machine interfaces in industrial robotics. Report for the U.S. Army Human Engineering Laboratory. Alexandria, VA: Essex Corporation.

Petersen, R.J., Hester, T., and Yorchak, J.P. (1984). Examine: An expert system to mediate human-computer dialogs. In Proceedings of the Human Factors Society--28th Annual Meeting. Santa Monica, CA: Human Factors Society.

Pew, R.W. (1988). Human factors issues in expert systems. In M. Helander (Ed.), Handbook of human-computer interaction. Amsterdam: North-Holland.

Phillips, E.R., Eike, D.R., and Fleger, S.A. (1986). Human factors issues in designing explanation facilities for expert trouble-shooting systems. In Proceedings of the Human Factors Society--30th Annual Meeting. Santa Monica, CA: Human Factors Society.

Shepard, R.N. (1987). Toward a universal law of generalization. Science, 237, 1317-1323.

Sheridan, T.B. (1987). Teleoperation, telepresence, and telerobotics: Research needs for space. In T.B. Sheridan, D.S. Kruser, and S. Deutsch (Eds.), Human factors in automated and robotic space systems: Proceedings of a symposium. Washington, DC: Committee on Human Factors, National Research Council.

Sheridan, T.B. and Hennessy, R.T. (Eds.) (1984). Research and modeling of supervisory control behavior. Washington, DC: National Academy Press.

Smith, T.J. and Smith, K.U. (1985). Cybernetic factors in motor performance and development. In D. Goodman, R.B. Wilberg, and I.M. Franks (Eds.), Differing perspectives in motor learning, memory, and control. Amsterdam: North-Holland.

Spain, E.H. (1987). Assessments of maneuverability with the teleoperated vehicle (TOV). Presentation at the Fourteenth Annual Symposium of the Association of Unmanned Vehicle Systems, Washington, DC.

Stark, L. (1987). Telerobotics for the evolving space station: Research needs and outstanding problems. In T.B. Sheridan, D.S. Kruser, and S. Deutsch (Eds.), Human factors in automated and robotic space systems: Proceedings of a symposium. Washington, DC: Committee on Human Factors, National Research Council.

Thiele, A.W. (1987). Sensor integrated gripper system. In Proceedings of the International Topical Meeting on Remote Systems and Robotics in Hostile Environments. La Grange Park, IL: American Nuclear Society.

Wise, J.A. and Rosenberg, E. (1988). The effects of interior treatments on performance stress in three types of mental tasks. CIER Technical Report 002-02-1988. Allendale, MI: Grand Valley State University.

Figure 1. Human factors involvements in robotics, supervisory control, expert systems, and telepresence.

HUMAN FACTORS INVOLVEMENTS

TASK ANALYSIS
TASK ALLOCATION
INTERFACE DESIGN, HARDWARE
INTERFACE DESIGN, SOFTWARE
TEST/EVALUATION
SAFETY
JOB PERFORMANCE AIDS
TRAINING
PERSONNEL REQUIREMENTS
TEAMS
COMMUNICATIONS
SURVEY/SELF-REPORT
EXPERIMENTATION
PLANNING/GOAL SETTING
DECISION-MAKING
REAL-TIME CONTROL
MAINTENANCE
APPLICATIONS PROGRAMMING
FEEDBACK
PRESENCE
GENERALIZATION
WORKLOAD
TASK COMPLEXITY
TASK PREDICTABILITY

ELECTRONIC DATA GENERATION AND DISPLAY SYSTEM

Jules Wetekamm
Boeing Aerospace Operations
Cocoa Beach, FL 32932-0220

Abstract

The Electronic Data Generation and Display System (EDGADS) is a "field tested" paperless technical manual system. The authoring provides subject matter experts the option of developing procedureware from digital or hardcopy inputs of technical information from text, graphics, pictures, and recorded media (video, audio, etc.).

The display system provides multi-window presentations of graphics, pictures, animations, and action sequences with text and audio overlays on a high resolution color CRT and monochrome portable displays.

The database management system allows direct access via hierarchical menus, keyword name, ID number, voice command or touch screen pictorial of the item (ICON). It contains operations and maintenance technical information at three levels of intelligence for a total system.

In 1985, the Kennedy Space Center Space Station Logistics Systems Office selected BAO (Boeing Aerospace Operations) to analyze, assess and develop the technical data and documentation system requirements to support O&M (Operations and Maintenance) activities for Space Station on-orbit and ground operations. Our analysis of on-going related programs revealed that traditional paper-intensive O&M technical documentation techniques are expensive, time consuming to produce and maintain, and difficult to use in a micro-G environment. It was projected (as shown in **Figure 1**) that the Space Station Program technical information would exceed 1.5 million pages and require more than 75 thousand pages to provide autonomous support for on-orbit activities. The characteristics of a paper documentation system is, at best, cumbersome. It requires: (1) space in the operations area, (2) time consuming accessibility and update procedures, (3) multi-page reference trails, and (4) high cost to acquire and place in orbit.

Example: It would cost 44 million 1985 dollars to deliver the 75 thousand page file system to low earth orbit.

Figure 1
HARD COPY TECHNICAL DOCUMENTATION
REQUIREMENTS SUMMARY

	ASSEMBLY	REPAIRING & INSPECTION	MAINTENANCE	OPERATIONS	REPAIR & MAINTENANCE	TRAINING	VERIFICATION	DOCUMENTS	PAGE
GROUND	200	20	40	120	20	20	20	—	—
SPACE	110	110	1200	1000	1200	1000	1000	8000	—
TOTAL PAGE	310	130	1240	1120	1220	1020	1020	8000	32240
UNIQUE	200	20	40	120	20	20	20	—	—
SPACE	110	110	1200	1000	1200	1000	1000	8000	—
TOTAL PAGE	310	130	1240	1120	1220	1020	1020	8000	32240

NOTE: (1) DOES NOT INCLUDE BOX, PACK & SHIP WPI, WPI & WPI USE, EXPERIMENTS, OTV, PLATFORM OR CONSUMABLE STORAGE, DESIGN, ETC., COMPUTER PROGRAM DOC., ILLUSTRATED PARTS LIST, FACILITIES

(2) CONSIDERS INCORPORATION OF MAXIMUM BIT-RATE

(3) * SPACE STATION TOTAL TECHNICAL DOCUMENTATION APPROACHES 1.5 MILLION PAGES
* STS-1 REQUIRED APPROXIMATELY 8 MILLION PAGES DURING 30 MONTH PROCESS
* STS-2 AND TODAY * 28,000 PAGES
* STS-3 ENGINEERING, LOGISTICS, USE AND FACILITY * 1,800,000 PAGES

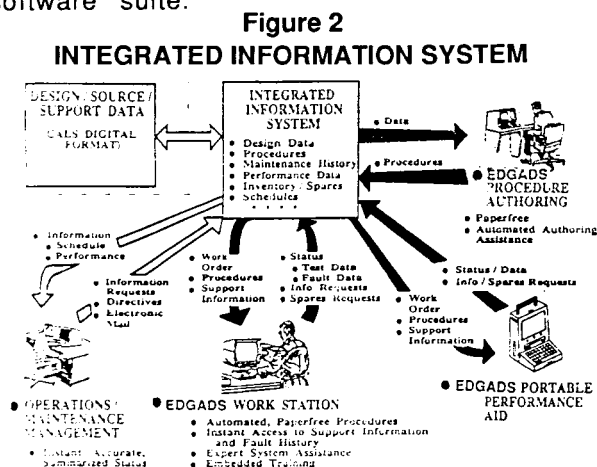
The unwieldly behavior and the inherent characteristics of paper are intolerable in intra-vehicular activities and impossible in extra-vehicular activities.

These initial analysis of the space station technical documentation investigations concluded:

- o An end-to-end "paperless" ETDS (Electronic Technical Documentation System) eliminates these undesirable inherent characteristics.
- o Current technology and existing hardware could support the requirements for developing and demonstrating the feasibility of a paperless technical information system.

ETDS CONCEPT

The BAO concept (shown in *Figure 2*) for the Space Station Program ETDS is to electronically acquire, author, compress, integrate and interface various forms of technical information from design, manufacturing and product support data bases for use in supporting operations and maintenance activities. The process will utilize computer-based digital, audio, video, laser and other electronic devices to provide the operator/user hands-free or eyes free selection and use of O&M technical information without the need for generating voluminous technical procedures and/or manuals in paper format. The ETDS is comprised of two major components: (1) a TIAS (Technical Information Acquisition Standard, and (2) an EDGADS (Electronic Data Generation and Display System) hardware and software suite.



I. Technical Information Acquisition Standard

The TIAS serves as a guide for prescribing and managing the digitized formats, media, content, and automated interchange of technical information between designers, authors, editors and consumers to support space station operations, maintenance, logistics support, verification, validation, and training functions. It incorporates applicable industry and government accepted standards and provides for handling non-digitized data. It presents explicit examples of User Video Displays developed from digitized formats (arrangements) of design and manufacturer source materials. Draft Version 4 was submitted to NASA KSC SS-LSO for staffing on 4/10/88.

II. Electronic Data Generation and Display System

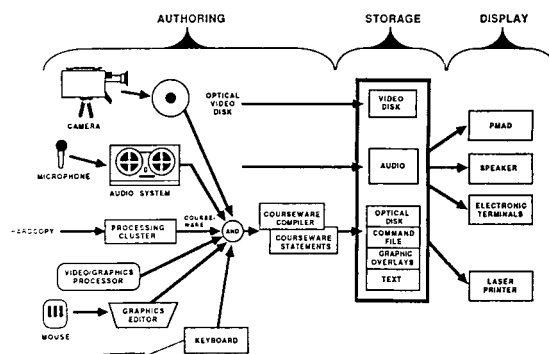
The approach for developing the EDGADS is shown in *Figure 3*.

EDGADS is defined by the following requirements:

AUTHORING SYSTEM

The authoring system will provide computer aided authoring of technical information by subject matter experts from hardcopy (text, graphics, pictures) and recorded media (video, audio).

Figure 3
EDGADS APPROACH



**ORIGINAL PAGE IS
OF POOR QUALITY**

ORIGINAL PAGE IS OF POOR QUALITY

DISPLAY SYSTEM

The display system will provide multi-window presentations of graphics, pictures, animations, action sequences with text statements and audio overlays on high resolution color CRT. It will include a print option and a monochrome portable device.

STORAGE SYSTEM

The storage system will be an Optical disk (WORM) drive controlled by work station and transparent to input/output hardware.

DATA BASE MANAGEMENT SYSTEM

The DBMS will be structured to allow direct access via hierarchical menus, keyword, name, ID number, voice command or touchscreen pictorial of the item.

It will contain operations and maintenance information at three levels of intelligence for a total system.

APPLICATIONS SOFTWARE

The applications software will provide for the following:

- a) Automated data presentations
- b) Switching between operations instructions and related maintenance instructions
- c) Sequence termination and restart at same point
- d) Maintaining date, time and personnel ID log
- e) Intelligence level selection and change at anytime
- f) QA control functions
- g) Out-of-sequence branching.

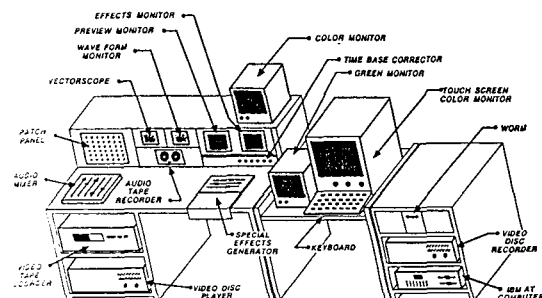
EDGADS user requirements:

- o Hands-free operation
- o Immediate access to Support Data Bases
- o Automatic verification of Support Test Equipment
- o Authenticates users
- o Prevents accidental deletion of base-line data (WORM)
- o Allows authorized amendments to accommodate unique or field required modifications (change management)

- o Allows authorized amendments to accommodate unique or field required modifications (change management)
- o Automatic update of historical and logistics support files
- o Date, time and personnel ID log by task or procedural step (print on demand)
- o Allows user to develop new or special test routines
- o Contains all processing and control forms
- o Includes embedded User's Guide.

In February of 1986, Boeing Aerospace agreed to supplement the funding of an on-going IR&D project (PBA-502) to develop a proof of concept demonstration on an operational system to prove feasibility of the EDGADS implementation aspects. Part of the Product Support Development Station hardware (as shown in **Figure 4**) and software was modified and dedicated to this effort. The project was designated as BIDS (Boeing Intelligent Data System).

Figure 4
BIDS DEVELOPMENT STATION



BIDS IR&D Project Objectives

The overall objectives of the Boeing Intelligent Data System (BIDS) project for 1986 were to:

1. Develop the EDGADS conceptual design to perform computer based authoring, storage and presentation functions for technical data to support operations, maintenance, training and inventory management activities for Space Station and other related DOD programs.

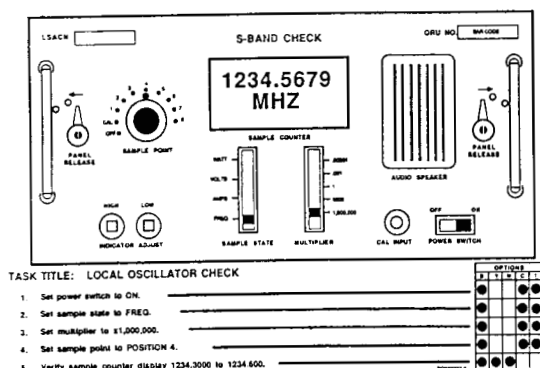
2. Integrate a concept demonstrator consisting of the BAC Training Equipment Systems (TES), baseline Computer Based Instructional (CBI) delivery station.
3. Procure, develop and integrate the various hardware and software elements to support a demonstration.
4. Develop and demonstrate typical subsystem maintenance scenarios which will reflect a new documentation and support concept for the Space Station program.
5. Provide support for demonstrations and meetings.
6. Document activities of above tasks.

All of these objectives were met.

EDGADS Proof of Concept Demonstration

The EDGADS conceptual design was successfully demonstrated to NASA, Navy and AF personnel on June 3, 1986 in the BCAC 757 Trainer Simulator Facility. The demonstration was a paperless audio and video presentation of the 757 procedural steps to provide a moderately trained technician to perform the functions for resolving a malfunction in the 757 right engine generator system and return it to proper operation. A video documentary of the demonstration is available for VHS viewing on request.

**Figure 5
EDGADS DISPLAY**



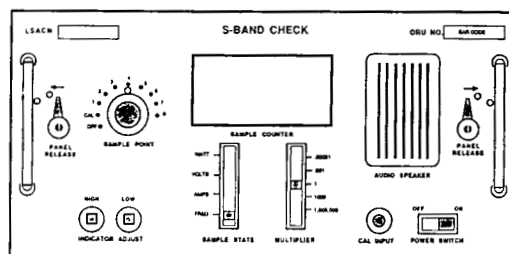
**ORIGINAL PAGE IS
OF POOR QUALITY**

ETDS End-to-End Paperless Procedure Development

The end-to-end paperless procedure development, as shown in Figure 2, was simulated by providing examples of engineering source data and a composite of the user's desired display format to the EDGADS author. These examples are shown in Figures 5, 6 and 7. When the ETDS is implemented, the EDGADS procedure authoring station could acquire these files electronically. The **EDGADS Display** (as shown in Figure 5) is a combination of the **EDGADS Input File** (shown in Figure 6) and the **Local Oscillator Check** elements of the other **Input File** (Figure 7). It also provides the author explicit instructions for implementing the user requirements for satisfying the qualification testing protocols, QA actions and operator interface techniques.

The most significant of these instructions involves the "options" matrix which is explained as follows:

**Figure 6
EDGADS INPUT**



Options Matrix Codes

B - executes a browse capability

Y - indicated actions were successful

N - indicated action not successful; automatically branches to trouble-shooting menu; must be cleared to continue

C - indicates action complete; proceeds to next step

I - indicated action incomplete; can browse, quit, or do other task out of sequence; flags control system that test is incomplete.

The front panel drawing was manipulated to produce the O&M user displays for performing the selected tasks/activities as desired. The highlighted/ animated frames are maintained in the O&M graphics file and can be accessed by **LSACN, ORU No., NOUN NAME** as indicated on the drawing.

Figure 7
EDGADS INPUT

		S-BAND CHECK															
TASKS	ACTIVITIES	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF	POWER ON	POWER OFF
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LOCAL OSCILLATOR CHECK		1	2														
VOICE QUALITY CHECK		1	4														
RF POWER CHECK		1	6		2		3										

IUS Ground Power Control Rack Demonstration

Although the 757 demonstration proved the feasibility of the EDGADS concept, it did not address the ramifications of a real NASA/DOD operational environment; such as, quality verification, contractual delivery requirements, monitoring protocols, configuration management (procedure), and performance authorization. Therefore, it was decided to utilize the Air Force Eastern Launch Site (ELS) Inertial Upper Stage (IUS) off-line facility to demonstrate the paperless concept in the operational environment on the Ground Power Control Rack.

The Ground Power Control Rack (GPCR) provides power at the various voltage levels required for operating and calibrating the IUS flight systems during assembly and certification testing at the ELS.

IUS GPCR Demonstration Features

- o Operates with both base and portable computers
- o Accepts keyboard, voice and touch inputs
- o Displays test document accompanied by video, graphic and audio aids
- o Verifies (barcode) personnel and equipment
- o Automatically logs all test data
 - o Test progress (time, date, status, etc. for each step)
 - o Equipment use
 - o Required buy-off/signatures
 - o Pickups, UER's, etc.
- o Retrieves and displays
 - o Drawings ("A" size in demo)
 - o Specifications
 - o Historical data (related UER's, test logs, etc.)

Although the entire procedure was automated, only selected segments were performed in four demonstration sessions which were viewed by Air Force (uniform and civilian), Navy civilians, and NASA personnel.

The Demonstration Scenario included: a brief introduction to the **C**heck-out **E**xecutive **S**ystem (**CHEX**); **S**ystem **S**tartup sequence; general notes from IRSO; **A**C **F**unction **P**ower **T**est and 28V DC **F**PT; **B**rowse **T**est **L**og; **G**o **O**ut-of-**S**equence to **W**rap **A**round **T**est; simulate a fault and fill out a Pickup; view related specifications; begin repeat of **W**RAP **T**est; perform clean-up sequence and "print" buy-off sheet.

The video documentary of the as-run demonstration clearly shows the superiority of paperless prompting. It also requires 40% less time to perform and eliminates 80% of the pre-test and post-test activities imposed by paper-driven systems.

AUTHOR INDEX

Abd-Allah, Mahmoud A.	353	Fesq., Lorraine	533
Amell, John R.	233	Fink, Pamela K.	59
Andary, J. F.	391	Ford, Donnie	47
Anderson, Audie	47	Fouse, Scott	177
Andrews, Alison	73	Flaherty, Doug	105
Arens, Yigal	205	Frank, Michael	105
Aspinall, John G.	153	Fracker, Capt. Martin L.	227
Avula, Xavier J.R.	403		
		Garin, John	411
Bachert, Robert F.	527	Genesereth, Mike	299
Baffes, Paul	383	Grissom, William A.	353
Baltzley, Dennis R.	313	Gyamfi, Max	471
Bettinger, Keith E.	245		
Brady, Mike	47	Haralick, Robert M.	371
Brink, James R.	41	Harris, Randall L. Sr.	221
Brown, H. Benjamin, Jr.	429	Hebert, Martial	373
Brown, Max	343	Heggestad, Harold M.	53
Bryoun, Jung	245	Heindel, Troy A.	51
Bylander, Tom	137	Hinkal, S. W.	391
		Hino, James	167
Cai, Chunsheng	343	Holt, Robert H.	509
Cannon, David	287	Homeier, Peter	187
Ciccarelli, Eugene	193	Hosein, Marc	513
Chang, C. L.	131	Hubert, Capt. James A.	303
Chatterjee, Chanchad	369		
Cheatham, J. B.	467	Jaap, John	1, 7
Churchill, Philip J.	481	Jared, David	511
Clark, Margaret M.	495	Jennings, Von Ayre	411
Cleghorn, Timothy	383, 467	Johnson, Sally C.	123
Colle, Herbert A.	233	Johnson, William B.	267
Comstock, J. Raymond, Jr.	221	Jones, Marshall B.	313
Conway, Lynn	71	Josephson, John R.	149
Crabb, T.	397		
		Kaleps, Ints	403
Davis, Elizabeth	1	Kamil, Hasan	499
Davis, Gloria	515	Kan, Edwin P.	437
Dickinson, Jacob L.	105	Kanade, Takeo	373, 421
Dickinson, William J.	81	Kelly, Capt. Brian	301
Dobes, Zuzana	245	Kelly, Frederick A.	339
Duffie, N.	397	Kennedy, Robert S.	313
Duke, Eugene L.	107	Khosia, Pradeep	421
Duncan, Phillip C.	267	King, James A.	141
		Kruchten, Robert J.	161
Evers, Ken H.	527	Kweon, In So	373
Ewry, Michael E.	233		
		Lakin, Fred	287
Faymen, Karl A.	501	Lamont, Gary B.	179
Feiner, Steven	253	Le, Thach	187
Feliu, Vincente	429	Leifer, Larry	287

Litt, Johnathan	503
Little, Arthur D.	481
Lu, Li	343
Lum, Jr., Henry	153
Madison, Robin M.	51
Manouchehri, Davoud	495
Marcus, Beth A.	481
Mark, Bill	299
Matteo, Joseph	411
McFarland, Robert Z.	51
Miller, Lawrence	205
Mitchell, Paul	21
Morera, Osvaldo	237
Morris, Keith	517
Mueller, Stephen J.	535
Mullins, Capt. Barry	31
Muratore, John F.	51
Murphy, Terri B.	51
Nardi, Bonnie	193
Neal, Jeanette G.	245
Norton, Jeffrey E.	267
Nugent, Richard O.	171
Nygren, Thomas E.	237
Obergefell, Louise	403
Olson, Judith Reitman	293
Oppenheim, Irving J.	487
Palmer, Karol K.	59
Parsons, H. McIlvaine	541
Patrick, Clint	47
Petrosky, Lyman J.	487
Pickl, William G.	309
Pope, Alan T.	221
Rasmussen, Arthur N.	51
Rattan, Kuldip S.	429
Regian, J. Wesley	273
Riesback, Christopher K.	289
Ross, Charles L.	167
Rueter, Henry H.	293
Russo, Carol J.	69
Schlossberg, Jon	299
Schmitz, Donald	421
Scholz, Arthur L.	81
Schroeder, Joyce D.	451
Schultz, Roger D.	87
Schur, Anne	259
Scoggins, Terrell	525

Scruggs, Jeffrey L.	461
Shakley, Donald J.	179
Shapell, Roger	327
Shapiro, Linda G.	361
Sheu, Shih Ying	71
Shields, Jr., Nicholas	329
Shrobe, Howard E.	153
Singh, Narinder	299
Sivard, Cecilia	287
Smith, Jeffrey H.	471
Smith, Randy L.	321
Sneckenberger, John E.	335
Sondheimer, Norman	205
Soni, A.H.	343
Stachowitz, R. A.	131
Stacy, Kenneth L.	7
Sternberg, Stanley R.	369
Storey, Paul	41
Stuart, Mark A.	321
Sudkamp, Thomas	143
Sullivan, Joseph W.	299
Sullivan, Mark	287
Sundberg, Gale R.	501
Szatkowski, Gerard P.	87
Teeter, R.	397
Thielman, Carol Y.	245
Todd, Capt. Wayne	161
Toptani, Richard	369
Truszkowski, Walter F.	279
Tucker, Richard W.	171
Tyler, Sherman W.	219
Vaishl, Ashok K.	499
Volkmer, Kent	471
Volz, Richard A.	71
Walker, Michael W.	71
Wambaugh, John	287
Wang, Caroline	47
Wang, Lui	383
Watzin, J. G.	391
Weeks, David J.	25
Weiland, P. L.	467
Westinghouse	487
Wetekamm, Jules	553
Whitaker, Leslie	141
White, Carl L.	353
Wilcox, Brian H.	445, 457
Williams, Linda J.F.	95
Wong, Joe	449
Wu, C. K.	467

Yoshikata, Kazuk	335
Zeanah, Hugh	47
Zik, J.	397
Zimmerman, Wayne	471
Zweben, Monte	15

REPORT DOCUMENTATION PAGE

1. Report No. NASA CP-3019		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88)				5. Report Date November 1988	
				6. Performing Organization Code	
7. Author(s) Sandy Griffin, editor/compiler				8. Performing Organization Report No. S-585	
				10. Work Unit No.	
9. Performing Organization Name and Address Lyndon B. Johnson Space Center Houston, Texas 77058				11. Contract or Grant No.	
				13. Type of Report and Period Covered Conference Publication	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration, Washington, D.C. 20546 U.S. Air Force, Washington, D.C. 23304 Wright State University, Dayton, OH 45324				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract Papers presented at the Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88), hosted by Wright State University at Dayton, Ohio, on July 20, 21, 22, and 23, 1988, are documented herein. During the 4 days, approximately 100 technical papers were presented by experts from NASA, the USAF, universities, and technical companies. Panel discussions on Human Factors, Artificial Intelligence, Robotics, and Space Systems were held but are not documented herein. Technical topics addressed included knowledge-based systems, human factors, and robotics.					
17. Key Words (Suggested by Author(s)) knowledge acquisition flight crew life support, protection, and intelligent tutoring teleoperators emergency escape task planning, reasoning, human factors engr knowledge-based systems verification, and robotics artificial intelligence validation space structures manipulation and effectors autonomous navigation sensing and perception CLIPS				18. Distribution Statement Unclassified - Unlimited Subject Category: 59	
19. Security Classification (of this report) Unclassified		20. Security Classification (of this page) Unclassified		21. No. of pages 564	
				22. Price A24	